

UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA E INFORMÁTICA
CONCEPCIÓN - CHILE



**“PROYECTO STUTRABAJO: DESARROLLO DEL
BACKEND DE PLATAFORMA DE EMPLEOS PARA
ESTUDIANTES DE EDUCACIÓN SUPERIOR”**

SEBASTIÁN ALEXANDER POBLETE GODOY

**MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO EN INFORMÁTICA**

Profesor Guía: Cristian Antonio Lara Valenzuela

Julio - 2025



CONSTANCIA DE VALIDACIÓN Y CONFIDENCIALIDAD DE MONOGRAFÍA A REPOSITORIO ACADÉMICO

1.- IDENTIFICACIÓN DEL TRABAJO ACADÉMICO

Tipo de monografía (marcar una opción): ☒ Memoria o trabajo de título; ☐ Tesis de Postgrado;

Título del trabajo: PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACIÓN SUPERIOR

Nombre del candidato(a): SEBASTIÁN ALEXANDER POBLETE GODOY

Carrera / Grado: INGENIERIA EN INFORMÁTICA

Campus: Concepcion ; Departamento: INFORMÁTICA

2.- VALIDACIÓN DEL PROFESOR GUÍA/DIRECTOR DE TESIS

Yo, Cristian Antonio Lara Valenzuela, en mi calidad de profesor(a) guía/director(a) del trabajo académico mencionado anteriormente **DEJO CONSTANCIA** que:

- He revisado esta versión del documento y corresponde a la versión final aprobada del trabajo.
- El trabajo cumple con los requisitos académicos y de formato establecidos por la institución

3.- EVALUACIÓN DE CONFIDENCIALIDAD POR PROPIEDAD INDUSTRIAL

☒ El trabajo **NO contiene información que amerite confidencialidad** y puede ser publicado de inmediato en repositorio con acceso abierto.

☐ El trabajo **CONTIENE** información con potenciales implicancias de propiedad industrial o intelectual y requiere un periodo de confidencialidad (embargo) por:

☐ 6 meses; ☐ 12 meses; ☐ 2 años; ☐ 3 años; ☐ 5 años; ☐ 10 años

Fundamentación de la necesidad de confidencialidad (obligatorio si se solicita embargo):

4.- FIRMAS

Profesor(a) guía o director(a) de memoria o tesis:

Fecha: 24-07-2025

; Firma:

CRISTIAN LARA
VALENZUELA

Firmado digitalmente por
CRISTIAN LARA VALENZUELA
Fecha: 2025.07.24 18:19:41
-04'00'

Estudiante o Candidato(a):

Fecha: 24-07-2025

; Firma:

Este formulario debe ser insertado como página 2 de la memoria o tesis, completado y firmado por estudiante y profesor(a) antes de la entrega en portal PRISMA de Biblioteca USM.

DEDICATORIA

Dedico este trabajo, en primer lugar, a mi familia, por ser mi mayor pilar de apoyo y por su amor incondicional que me ha sostenido en los momentos más desafiantes. A mis padres, por enseñarme el valor del esfuerzo, la perseverancia y la fe en mis capacidades; a mis hermanos, por su ánimo constante y su compañía en este camino.

También dedico este proyecto a mis compañeros de equipo, con quienes compartí retos, aprendizajes y logros. Su dedicación y compromiso fueron fundamentales para hacer realidad esta propuesta.

Finalmente, dedico este logro a todas las personas que han creído en mí, quienes con su apoyo y confianza me inspiraron a dar lo mejor de mí mismo. Este trabajo representa el esfuerzo colectivo y las enseñanzas de todos aquellos que me han acompañado en este viaje.

Gracias a todos por ser parte de este sueño hecho realidad.

AGRADECIMIENTOS

Quiero expresar mi más profundo agradecimiento a todas las personas que, de una forma u otra, hicieron posible la realización de este proyecto y contribuyeron a mi desarrollo personal y profesional durante este proceso.

Primero, quiero expresar mi más sincero agradecimiento a Dios, por guiarme a lo largo de este proceso, brindándome la sabiduría, el conocimiento y la fortaleza necesarios para completar este proyecto con éxito.

Agradezco a mi familia por su amor incondicional, apoyo constante y comprensión durante todo este camino. Su confianza en mis capacidades y su motivación en los momentos más desafiantes han sido mi mayor fortaleza.

A mis compañeros de trabajo en el desarrollo de este proyecto, quiero darles las gracias por su dedicación, colaboración y profesionalismo. El esfuerzo conjunto y las ideas compartidas no solo permitieron construir una solución innovadora, sino también crear un ambiente de aprendizaje y crecimiento mutuo que valoraré siempre.

Finalmente, agradezco a mis profesores y mentores, quienes con su guía y conocimiento me han ayudado a superar obstáculos y alcanzar mis metas académicas. Este trabajo es el resultado de sus enseñanzas y su constante motivación para enfrentar nuevos desafíos.

A todos ustedes, gracias por ser parte de este importante logro en mi vida. Este trabajo no hubiera sido posible sin su apoyo, inspiración y confianza.

RESUMEN

Este proyecto consta del desarrollo de un portal de empleos de corto periodo de tiempo para estudiantes de Educación Superior en Chile, con el objetivo de proporcionarles oportunidades laborales que les permitan generar ingresos para cubrir necesidades básicas sin interferir con sus responsabilidades académicas. La plataforma fue construida mediante un backend robusto basado en una arquitectura de microservicios desplegada en Google Cloud Platform (GCP), utilizando tecnologías como Django REST Framework para la implementación de APIs RESTful y MySQL para la gestión de bases de datos. El backend desarrollado garantiza un rendimiento eficiente, escalabilidad modular y tiempos de respuesta adecuados, facilitando la publicación, postulación y gestión de empleos temporales. Esta solución contribuye significativamente a reducir la deserción estudiantil por motivos económicos y a mejorar la calidad de vida de los estudiantes, permitiéndoles equilibrar sus responsabilidades académicas y laborales.

ABSTRACT

This project consists of the development of a short-term job portal for higher education students in Chile, with the objective of providing job opportunities that allow them to generate income to cover basic needs without interfering with their academic responsibilities. The platform was built using a robust backend based on a microservices architecture deployed on Google Cloud Platform (GCP), utilizing technologies such as Django REST Framework for the implementation of RESTful APIs and MySQL for database management. The developed backend ensures efficient performance, modular scalability, and adequate response times, facilitating the posting, application, and management of temporary jobs. This solution significantly contributes to reducing student desertion for economic reasons and improving students' quality of life by enabling them to balance their academic and work responsibilities.

GLOSARIO

- **API:** *Application Programming Interface* (Interfaz de Programación de Aplicaciones).
- **CPU:** *Central Processing Unit* (Unidad Central de Procesamiento).
- **DER:** *Diagrama Entidad Relación*.
- **GCP:** *Google Cloud Platform* (Plataforma en la Nube de Google).
- **GKE:** *Google Kubernetes Engine* (Motor de Kubernetes de Google).
- **HTTP:** *Hypertext Transfer Protocol* (Protocolo de Transferencia de Hipertexto).
- **IDE:** *Integrated Development Environment* (Entorno de Desarrollo Integrado).
- **JVM:** *Java Virtual Machine* (Máquina Virtual de Java).
- **RUN:** Rol Único Nacional.
- **SIES:** *Servicio de Información de Educación Superior*.
- **SQL:** *Structured Query Language* (Lenguaje de Consulta Estructurada).
- **VS Code:** *Visual Studio Code* (Editor de Código de Microsoft).
- **XML:** *Extensible Markup Language* (Lenguaje de Marcado Extensible).

INDICE DE CONTENIDOS

RESUMEN.....	4
ABSTRACT	4
INDICE DE FIGURAS.....	10
INTRODUCCIÓN	11
CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA	12
1.1 Organización y contexto de trabajo	12
1.2 Actores o usuarios involucrados.....	13
1.3 Justificación y motivación.....	13
1.4 Competencia.....	14
1.5 Ventajas	15
1.6 Objetivos.....	16
1.6.1 Objetivo general	16
1.6.2 Objetivos específicos	17
CAPÍTULO 2: MARCO CONCEPTUAL	18
2.1 Conceptos Técnicos	18
2.1.1 Arquitectura de Microservicios	18
2.1.2 Kubernetes.....	18
2.2 Metodologías.....	19
2.2.1 Desarrollo Ágil (Scrum)	19
2.3 Herramientas	19
2.3.1 React y Vite	19
2.3.2 Google Cloud Platform (GCP)	19
2.3.3 Visual Studio Code	20

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA
ESTUDIANTES DE EDUCACION SUPERIOR

2.3.4	Django REST Framework	21
2.3.5	MySQL.....	21
2.4	Tecnologías Especificas.....	21
CAPÍTULO 3: PROPUESTA DE SOLUCIÓN		22
3.1	Descripción General de la Solución	22
3.2	Descripción Back-End	22
3.2.1	Introducción	22
3.3	Descripción General Back-End.....	22
3.3.1	Diseño Basado en Microservicios	22
3.3.2	Beneficios Clave de la Arquitectura en la Escalabilidad y el Mantenimiento	23
3.3.3	Criterios de Partición de Microservicios.....	24
3.3.4	Independencia y Despliegue.....	24
3.3.5	Resiliencia de Comunicación	25
3.3.6	Despliegue en Google Cloud Platform (GCP)	25
3.3.7	Componentes Principales del Sistema	25
3.4	Microservicios y Funcionalidad	26
3.5	Diseño de Base de Datos	29
3.5.1	Selección de SGDB	29
3.5.2	Users Database	29
3.5.3	Jobs Database	30
3.5.4	Chat Database.....	30
3.5.5	Normalización y Rendimiento	32
3.5.6	Consistencia y Relaciones	33
3.5.7	Índices y Rendimiento de Consultas.....	33

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN	34
4.1 Introducción	34
4.2 Metodología de Validación.....	34
4.2.1 Identificación de Objetivos.....	34
4.2.2 Pruebas Realizadas	35
4.2.2.1 Pruebas Funcionales:.....	35
4.2.2.2 Pruebas de Integración:.....	35
4.2.2.3 Pruebas de Carga y Rendimiento:	35
4.2.2.4 Validación General:	35
4.2.3 Herramientas Utilizadas	35
4.2.4 Diseño de los Casos de Prueba	36
4.2.5 Métricas de Evaluación.....	37
4.3 Pruebas Funcionales.....	37
4.3.1 Alcance.....	37
.3.2 Herramientas Utilizadas	37
.3.3 Metodología	38
.3.4 Resultados	39
.3.5 Métricas Clave	41
.3.6 Conclusiones.....	41
.4 Pruebas de Integración.....	42
4.4.1 Alcance.....	42
4.4.2 Herramientas Utilizadas	42
4.4.3 Metodología	42
4.4.4 Casos de Prueba	43

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA
ESTUDIANTES DE EDUCACION SUPERIOR

4.4.5	Resultados	44
4.4.6	Conclusiones	45
4.5	Pruebas de Carga y Rendimiento	45
4.5.1	Alcance.....	45
4.5.3	Metodología	46
4.5.5	Métricas Clave	51
4.5.6	Conclusiones	52
4.6	Validación General.....	52
4.6.1	Alcance.....	52
4.6.2	Metodología	53
4.6.3	Escenarios de Validación	53
4.6.4	Resultados	55
4.6.5	Conclusiones	56
4.7	Conclusión	56
CAPÍTULO 5: CONCLUSIONES		58
5.1	Conclusión General.....	58
5.2	Cumplimiento objetivos específicos.....	58
5.3	Alcances del Trabajo	59
5.4	Limitaciones del Proyecto	59
5.5	Contribuciones y Aportes	59
5.6	Recomendaciones y Líneas Futuras.....	60
5.7	Reflexión Final	60
REFERENCIAS BIBLIOGRÁFICAS.....		61

INDICE DE FIGURAS

Figura 1: Arquitectura de StuTrabajo Fuente: Elaboración Propia. _____	26
Figura 2: Modelo Entidad Relación, separadas por base de datos Fuente: Elaboración Propia. _____	32
Se devolvió correctamente la información del usuario autenticado. Figura 3: Devolucion de Token y Informacion Usuario. Fuente: Elaboración Propia. _____	39
Figura 4: Creacion de Empleos y visualizacion de los Empleos. Fuente: Elaboración Propia. _____	40
Figura 5: Postulacion a Empleo Realizada y Visualizacion de la Aplicacion. Fuente: Elaboración Propia. _____	40
Figura 6: Creacion de Usuarios y Obtencion de los Datos. Fuente: Elaboración Propia. _	41
Figura 7: Flujo de datos entre Microservicios Empleos y Feed service. Fuente: Elaboración Propia. _____	43
Figura 8: Flujo de datos entre Microservicios Usuario, Empleos y Feed service. Fuente: Elaboración Propia. _____	44
Figura 9: Flujo de datos en Feed service. Fuente: Elaboración Propia. _____	44
Figura 10: Tabla de resultados escenario 1, 100 Usuarios. Fuente: Elaboración Propia. _	47
Figura 11: Tabla de resultados escenario 1, 200 Usuarios. Fuente: Elaboración Propia.	47
Figura 12: Tabla de resultados escenario 1, 500 Usuarios. Fuente: Elaboración Propia.	48
Figura 13: Tabla de resultados escenario 2, 100 Usuarios. Fuente: Elaboración Propia.	48
Figura 14: Tabla de resultados escenario 2, 200 Usuarios. Fuente: Elaboración Propia.	49
Figura 15: Tabla de resultados escenario 2, 500 Usuarios. Fuente: Elaboración Propia.	49
Figura 16: Tabla de resultados escenario 3, 100 Usuarios. Fuente: Elaboración Propia.	50
Figura 17: Tabla de resultados escenario 3, 400 Usuarios. Fuente: Elaboración Propia.	51
Figura 18: Tabla de resultados escenario 3, 500 Usuarios. Fuente: Elaboración Propia.	51

INTRODUCCIÓN

La deserción estudiantil en las Instituciones de Educación Superior en Chile es un problema poco discutido que afecta tanto a los estudiantes como a las instituciones. Diversos estudios realizados por el Servicio de Información de Educación Superior (SIES), el Ministerio de Educación (MINEDUC) y Psicomatrix indican que, para el año 2023, hay 1.341.439 estudiantes matriculados en programas de Pregrado, Posgrado y Postítulo. De estos, el 21% abandona sus estudios por motivos económicos, vinculados a la falta de recursos para cubrir necesidades básicas como alojamiento, alimentación, transporte, matrícula, arancel, entre otros. Este problema surge debido a la incompatibilidad entre las responsabilidades académicas y laborales, obligando a muchos estudiantes a abandonar sus estudios para poder trabajar. Además, aquellos que intentan equilibrar estudio y trabajo a menudo experimentan efectos negativos en su desarrollo académico y profesional.

Este proyecto aborda dicha problemática mediante el desarrollo de un portal de empleos de corto periodo de tiempo dirigido exclusivamente a estudiantes de Educación Superior, facilitando la búsqueda de trabajos temporales adecuados a sus necesidades y disponibilidad. La solución propuesta busca proporcionar ingresos a los estudiantes y apoyar su éxito académico y profesional a largo plazo.

El desarrollo se basó en un enfoque centrado en los usuarios del portal, iniciando con un análisis exhaustivo del problema para identificar los desafíos principales y las necesidades específicas de los estudiantes. Además, se realizó un estudio de las competencias existentes y se definieron claramente los objetivos y alcances del proyecto. En el aspecto técnico, se abordó específicamente el desarrollo del backend, implementando una arquitectura de microservicios desplegada en Google Cloud Platform (GCP), utilizando Django REST Framework para crear APIs RESTful y MySQL como sistema de gestión de bases de datos.

Posteriormente, el proyecto detalla la propuesta de solución, describiendo los elementos clave que integran la plataforma, e incluye un plan de validación que demuestra la efectividad y pertinencia de la solución para los usuarios. Finalmente, concluye con una reflexión sobre los aprendizajes adquiridos durante la ejecución del trabajo y analiza las implicaciones futuras para la continuidad y mejora de la plataforma.

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA

1.1 Organización y contexto de trabajo

El proyecto se desarrollará en el contexto de la plataforma **StuTrabajo**, una iniciativa destinada a brindar apoyo financiero a estudiantes de Educación Superior en Chile. La organización detrás de este proyecto está comprometida con la misión de reducir la deserción estudiantil por motivos económicos, facilitando a los estudiantes la posibilidad de generar ingresos que les permitan cubrir necesidades básicas como alojamiento, alimentos, transporte, matrícula, arancel, etc.

El proyecto se enmarca en varias líneas de la informática, destacándose el desarrollo de software, ciberseguridad, inteligencia artificial, aprendizaje automático, bases de datos, sistemas de información, experiencia de usuario, entre otras.

Situación actual

Actualmente, los estudiantes suelen recurrir a diferentes métodos para encontrar empleo, entre ellos:

- Portales de empleo genéricos: Utilizan plataformas que no están específicamente diseñadas para estudiantes y no siempre se ofrecen empleos con la flexibilidad necesaria.
- Redes sociales y preferencias: Se apoyan en redes sociales y recomendaciones de amigos o familiares para encontrar trabajos temporales, aunque esta estrategia no siempre resulta efectiva ni fiable.
- Ferias de empleo: Participación en ferias de empleo organizadas por las mismas instituciones o entidades externas, aunque las oportunidades son demasiado limitadas y no siempre se ajustan a sus necesidades de tiempo.
- Bolsas de trabajo universitarias: Algunas instituciones ofrecen bolsas de trabajo, pero estas suelen ser limitadas de alcance ya que existen pocos cupos para poder postular y no hay mucha diversidad de ofertas.

El proceso que los estudiantes siguen para encontrar empleos típicamente incluye:

- Búsqueda de empleo: buscan activamente empleos en múltiples plataformas, lo que puede ser un proceso largo y agotador.
- Aplicación y envío de CV: Una vez que encuentran la oportunidad, hacen envío de su currículum y esperar una respuesta, lo cual puede tardar días o incluso semanas.

- Entrevistas y selección: Si son seleccionados, pasan por entrevistas que pueden ser presenciales o virtuales, requiriendo una inversión adicional de tiempo.
- Negociación de horarios: Muchos estudiantes deben negociar los horarios con los empleadores para asegurarse de que no interfieran con sus clases, lo cual no siempre es posible y provoca que el estudiante pierda la oportunidad de trabajar.

1.2 Actores o usuarios involucrados

Aquellas actores o usuarios que están involucrados en esta problemática son principalmente los estudiantes de Instituciones de Educación Superior y empleadores de Chile.

Los estudiantes buscan empleos temporales y flexibles que les permitan generar ingresos sin interferir con sus estudios. Actualmente, enfrentan diversas dificultades, como la falta de flexibilidad en los empleos disponibles, la escasa disponibilidad de plataformas especializadas en empleos temporales, y el largo y agotador proceso de búsqueda de empleo. Además, la inseguridad sobre la confiabilidad de los empleadores y la calidad de los empleos es una preocupación constante. Según datos del Ministerio de Educación de Chile, hay aproximadamente 1.3 millones de estudiantes de Educación Superior en el país y según la OCDE (2019), cerca del 40% de los estudiantes chilenos trabajan mientras estudian, lo que representa alrededor de 520,000 estudiantes.

Por otra parte, tenemos a los empleadores, que incluyen empresas, organizaciones y particulares que ofrecen empleos temporales y flexibles. Estos empleadores enfrentan dificultades para encontrar estudiantes que se ajusten a las necesidades específicas del empleo, y el proceso de reclutamiento y selección puede ser largo y consume mucho tiempo. Además, la falta de información y confianza en la calidad de los candidatos es una barrera significativa. La encuesta de la Cámara de Comercio de Santiago (2021) reveló que un 35% de las empresas en Chile contrata estudiantes para empleos temporales, lo que representa un mercado considerable para una plataforma que facilite este proceso.

1.3 Justificación y motivación

Si en el corto o mediano plazo no se solucionan las dificultades que enfrentan los estudiantes de Educación Superior en Chile para encontrar empleos temporales y flexibles, las consecuencias podrían ser significativas tanto a nivel individual como social.

A nivel individual, uno de los impactos más críticos es la posible deserción. Los estudiantes que no pueden equilibrar sus necesidades económicas con sus responsabilidades académicas pueden verse obligados a abandonar sus estudios. Según datos del Ministerio

de Educación de Chile, la deserción universitaria en el país ya es un problema considerable, con tasas que oscilan entre el 20% y el 30% en algunas instituciones. Además, la continua preocupación por la falta de ingresos puede generar altos niveles de estrés y ansiedad entre los estudiantes. Este estrés no solo afecta su bienestar emocional, sino también su rendimiento académico, creando un ciclo negativo que es difícil de romper.

A nivel social, la deserción tiene un impacto directo en las instituciones de Educación Superior. Cuando los estudiantes abandonan sus estudios debido a dificultades económicas, estas instituciones sufren una reducción en sus tasas de retención y graduación, lo que afecta negativamente su reputación y acreditación. Las universidades, centros de formación técnica, institutos profesionales, dependen de las tasas de retención para mantener su financiación y apoyo gubernamental, y la deserción estudiantil puede resultar en menos recursos y oportunidades para todos los estudiantes.

Además, la falta de acceso a empleos temporales y flexibles también afecta a las instituciones de Educación Superior al limitar su capacidad para apoyar a sus estudiantes. Los estudiantes de bajos recursos, que son los más afectados por la necesidad de trabajar mientras estudian, pueden quedar en desventaja en comparación con sus pares que no enfrentan las mismas presiones económicas. Esto genera una brecha en el rendimiento académico y en las oportunidades de éxito profesional, lo que a su vez impacta negativamente en la misión educativa de estas instituciones.

Cada estudiante que no puede continuar su educación y no puede desarrollar su potencial completo representa una pérdida significativa para las instituciones de Educación Superior en términos de innovación y contribuciones al desarrollo académico. La fuga de talento también puede reducir la capacidad de estas instituciones para atraer financiamiento para investigación y desarrollo, así como para establecer colaboraciones con la industria y otros sectores. Por lo tanto, abordar estas dificultades es esencial no solo para el bienestar de los estudiantes, sino también para la sostenibilidad y crecimiento de las instituciones educativas en el país.

La motivación para llevar a cabo este proyecto se basa principalmente en contribuir a la reducción de deserción estudiantil causada por problemas económicos, equilibrar las responsabilidades académicas con lo laboral, reduciendo el estrés y mejorando el bienestar de los estudiantes y permitirles adquirir experiencia laboral valiosa, que mejorará sus perspectivas de empleo a largo plazo y su desarrollo profesional.

1.4 Competencia

En el contexto actual, existen varias soluciones y plataformas que intentan abordar el problema de la búsqueda de empleo para estudiantes de Educación Superior en Chile. Sin

embargo, estas plataformas presentan limitaciones que impiden una solución efectiva y adaptada a las necesidades específicas de los estudiantes. A continuación, se presentan algunas de las principales competencias y se analiza por qué la solución propuesta por StuTrabajo sería mejor.

- Plataformas como LinkedIn, CompuTrabajo y Laborum son ampliamente utilizadas para la búsqueda de empleo en general. Aunque estos portales ofrecen una gran cantidad de ofertas laborales, no están específicamente diseñados para estudiantes y, por lo tanto, no siempre proporcionan la flexibilidad necesaria. Los empleos disponibles en estos sitios suelen requerir un compromiso de tiempo completo o fijo, lo que no se ajusta a los horarios variables y exigencias académicas de los estudiantes. Además, estos portales no suelen ofrecer mecanismos para asegurar que las ofertas sean temporales o de corto plazo, lo cual es una necesidad clave para los estudiantes.
- Los estudiantes a menudo recurren a redes sociales y referencias personales para encontrar trabajos temporales. Aunque esta estrategia puede ser útil, es inherentemente informal y no garantiza la seguridad ni la calidad de los empleos. Las ofertas encontradas a través de estos medios pueden carecer de la estructura y formalidad necesarias, y no siempre proporcionan un entorno confiable y seguro para los estudiantes.
- En otros países, se han desarrollado plataformas especializadas que abordan problemas similares. Por ejemplo, en Estados Unidos y Europa, existen plataformas como Handshake y StudentJob que se centran en conectar a estudiantes con empleos temporales y flexibles. Estas plataformas han tenido éxito al ofrecer funcionalidades específicas como algoritmos de emparejamiento, perfiles dinámicos de estudiantes y sistemas de valoración de empleadores. Sin embargo, estas soluciones están adaptadas a los contextos y necesidades de sus respectivos países, y pueden no ser directamente aplicables al contexto chileno sin adaptaciones significativas.

1.5 Ventajas

StuTrabajo presenta varias ventajas que la hacen superior a las competencias mencionadas anteriormente:

- **Especialización en Estudiantes:** StuTrabajo está específicamente diseñada para atender las necesidades de los estudiantes de Educación Superior en Chile. Esto significa que todas las funcionalidades y características de la plataforma están adaptadas para proporcionar la flexibilidad y temporalidad necesarias.

- **Match Interactivo:** Utiliza algoritmos de emparejamiento que conectan automáticamente a los estudiantes con ofertas laborales que se ajustan a sus perfiles y disponibilidad. Esto reduce significativamente el tiempo y esfuerzo necesarios para encontrar un empleo adecuado.
- **Perfiles Dinámicos y Valoraciones:** Los estudiantes pueden crear perfiles detallados y recibir valoraciones de empleadores, lo que mejora la transparencia y confianza en el proceso de contratación. Los empleadores también se benefician al poder evaluar la calidad y adecuación de los candidatos antes de contratarlos.
- **Alertas y Notificaciones:** La plataforma envía alertas inmediatas para nuevas oportunidades de trabajo, asegurando que los estudiantes no pierdan ninguna oferta relevante. Esto es particularmente útil para estudiantes con horarios muy variables y ocupados.
- **Seguridad y Confiabilidad:** La plataforma garantiza un entorno seguro y confiable mediante la implementación de medidas de seguridad robustas y un sistema de verificación tanto para estudiantes como para empleadores.

Aunque existen varias alternativas en el mercado, ninguna aborda de manera integral y especializada las necesidades de los estudiantes de Educación Superior en Chile. StuTrabajo se posiciona como una solución superior al ofrecer una plataforma adaptada, funcional y segura, que facilita la conexión entre estudiantes y empleadores de manera eficiente y efectiva.

1.6 Objetivos

1.6.1 Objetivo general

Diseñar, implementar y validar un back-end basado en microservicios expuesto mediante un API Gateway y APIs RESTful construidas con Django REST Framework que emplee MySQL como motor relacional y se integre con Keycloak para autenticación federada.

El sistema debe:

- Gestionar de forma centralizada y segura los procesos de publicación, postulación, contratación y evaluación de empleos temporales para estudiantes chilenos.
- Mantener tiempos de respuesta promedio ≤ 500 ms con hasta 500 usuarios concurrentes, garantizando una disponibilidad $\geq 99,9$ % gracias a su despliegue contenerizado y orquestado en Kubernetes sobre GCP.
- Facilitar la evolución futura mediante una arquitectura modular y la documentación completa de sus APIs y pruebas.

1.6.2 Objetivos específicos

- **Levantamiento de requisitos y dominio.**
 - Analizar los procesos funcionales de StuTrabajo para identificar actores, casos de uso y requisitos técnicos del back-end y de la API.
- **Modelo de datos y diseño de servicios.**
 - Elaborar un DER normalizado y definir el particionamiento lógico en tres bases de datos.
 - Delimitar seis microservicios y sus contratos REST, aplicando principios de DDD y de escalabilidad diferenciada.
- **Construcción e integración.**
 - Implementar los microservicios con Django REST Framework, montar cada uno en contenedores Docker y exponerlos tras un API Gateway protegido por Keycloak y JWT.
- **Verificación de calidad.**
 - Desarrollar pruebas unitarias e integradas con el cliente de DRF y Postman.
 - Ejecutar pruebas de carga y estrés con Apache JMeter (100, 200 y 500 usuarios) verificando ≤ 500 ms de latencia promedio y > 98 % de éxito.
 - Documentar cobertura y resultados, destacando cuellos de botella detectados.

CAPÍTULO 2: MARCO CONCEPTUAL

Durante este capítulo, se describirán los fundamentos teóricos y conceptuales que sustentan la solución propuesta en este trabajo. Se abordan los conceptos técnicos, metodológicas y herramientas fundamentales, estableciendo una base sólida para la comprensión y desarrollo del proyecto.

2.1 *Conceptos Técnicos*

2.1.1 *Arquitectura de Microservicios*

La arquitectura de microservicios es un estilo de diseño software para la web de nuestro proyecto, donde los componentes de una aplicación se distribuyen en servicios pequeños, independientes y desplegados de forma autónoma. Cada microservicio tiene sus funciones específicas y que se comunican con los otros servicios a través de las APIs.

Ventajas:

- **Escalabilidad:** Permite la escalabilidad de los componentes según la demanda.
- **Mantenibilidad:** Facilita el mantenimiento y actualización de cada servicio sin afectar el resto de los componentes del sistema.
- **Despliegue Independiente:** Los microservicios pueden ser desplegado de forma independiente

2.1.2 *Kubernetes*

Google Kubernetes Engine (GKE) es un servicio de orquestación de contenedores gestionado por Google Cloud Platform (GCP) que automatiza la implementación, el escalado y la gestión de aplicaciones en contenedores. Basado en el proyecto de código abierto Kubernetes, GKE facilita la gestión de aplicaciones basadas en microservicios, proporcionando alta disponibilidad, escalabilidad automática y una integración fluida con otros servicios de GCP.

Características:

- **Orquestación de Contenedores:** Administra contenedores Docker, asegurando su correcta distribución y ejecución.
- **Escalabilidad Automática:** Ajusta automáticamente los recursos según la demanda.
- **Auto reparación:** Reinicia contenedores que fallan, reemplaza nodos y resuelve problemas de red.

2.2 Metodologías

2.2.1 Desarrollo Ágil (Scrum)

Scrum es una metodología ágil para la gestión y desarrollo de proyectos de software. Se basa en la formación de equipos autoorganizados y multifuncionales, trabajando en sprints cortos para entregar incrementos funcionales del producto.

Principios:

- **Iterativo e Incremental:** Desarrollo en ciclos cortos (sprints) con entregas parciales pero funcionales.
- **Colaboración:** Comunicación constante entre los miembros del equipo y las partes interesadas.
- **Adaptabilidad:** Capacidad para responder a cambios y ajustar el plan de trabajo según las necesidades emergentes.

2.3 Herramientas

2.3.1 React y Vite

React es una biblioteca de JavaScript para construir interfaces de usuario, desarrollada por Facebook. Vite es un entorno de desarrollo front-end que permite una configuración mínima y una compilación rápida, ideal para proyectos que utilizan React.

Ventajas:

- **React:** Componentes reutilizables, virtual DOM para rendimiento mejorado.
- **Vite:** Desarrollo rápido y eficiente, recarga en caliente.

2.3.2 Google Cloud Platform (GCP)

GCP es una suite de servicios en la nube que ofrece Google, proporcionando infraestructura, almacenamiento y servicios de machine learning. Es especialmente útil para proyectos que requieren escalabilidad y fiabilidad.

Componentes Clave:

- **Google Kubernetes Engine (GKE):** Para la orquestación de contenedores.
- **Google Cloud Storage:** Almacenamiento escalable y seguro.
- **Google Cloud SQL:** Bases de datos relacionales gestionadas.

2.3.3 Visual Studio Code

Visual Studio Code (VS Code) es un editor de código fuente desarrollado por Microsoft. Lo usan desarrolladores por su rendimiento, extensibilidad y soporte para múltiples lenguajes de programación. VS Code proporciona un entorno de desarrollo versátil que mejora la productividad y la eficiencia en la codificación.

Características Principales:

- **Extensibilidad:** VS Code tiene una vasta colección de extensiones que permiten a los desarrolladores agregar soporte para nuevos lenguajes, depuradores, integraciones con herramientas de desarrollo y servicios en la nube, entre otros. Esto permite personalizar el entorno de trabajo según las necesidades específicas del proyecto.
- **IntelliSense:** Proporciona autocompletado de código, resaltado de sintaxis y sugerencias de código basadas en el contexto, lo cual mejora la velocidad y precisión de la codificación.
- **Depuración Integrada:** Incluye herramientas de depuración que permiten establecer puntos de interrupción, inspeccionar variables, y realizar seguimiento del flujo de ejecución del código, todo dentro del editor.
- **Terminal Integrado:** VS Code incluye un terminal integrado, permitiendo a los desarrolladores ejecutar comandos de shell sin salir del entorno de desarrollo.
- **Editor Multilenguaje:** Soporte para una amplia variedad de lenguajes de programación y scripts, incluyendo pero no limitado a JavaScript, TypeScript, Python, C++, Go y Rust.

Ventajas de VS Code:

- **Ligero y Rápido:** VS Code es conocido por ser un editor ligero que se inicia rápidamente y consume menos recursos del sistema en comparación con otros entornos de desarrollo integrados (IDEs).
- **Multiplataforma:** Está disponible para Windows, macOS y Linux, proporcionando un entorno de desarrollo consistente en todas las plataformas.

- **Personalización:** Los desarrolladores pueden personalizar su entorno de trabajo utilizando temas, configuraciones personalizadas y atajos de teclado.
- **Comunidad Activa:** Una comunidad activa de desarrolladores contribuye regularmente con extensiones, temas y soluciones a problemas comunes, enriqueciendo el ecosistema de VS Code.

2.3.4 Django REST Framework

Django REST Framework Herramienta flexible para crear APIs RESTful robustas y escalables, facilitando una implementación rápida y segura del backend.

2.3.5 MySQL

MySQL es un sistema de gestión de bases de datos relacional estable y eficiente, ideal para manejar grandes volúmenes de datos, garantizando integridad y rendimiento.

2.4 Tecnologías Específicas

Las tecnologías específicas empleadas en el desarrollo del backend del proyecto incluyen:

- Django REST Framework
- MySQL
- Arquitectura de Microservicios
- Google Cloud Platform y Google Kubernetes Engine

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN

3.1 Descripción General de la Solución

El proyecto se centra en abordar la deserción estudiantil por razones económicas creando un portal de empleo temporal orientado específicamente a estudiantes de educación superior. Este portal pretende dar oportunidades de trabajo flexibles adaptadas a los horarios académicos de los estudiantes, permitiéndoles generar ingresos y continuar sus estudios.

3.2 Descripción Back-End

3.2.1 Introducción

El proyecto StuTrabajo tiene como objetivo desarrollar una plataforma de empleo que permita a estudiantes de educación superior encontrar trabajos temporales adaptados a sus horarios académicos. En este contexto, el desarrollo del Back-end es esencial, ya que proporciona la infraestructura necesaria para manejar la lógica de negocio, garantizar la seguridad de los datos y orquestar la comunicación entre los diferentes componentes del sistema.

El diseño del Back-end está orientado a satisfacer requisitos clave de rendimiento y escalabilidad. Para asegurar la capacidad del sistema, se ha planificado que este pueda manejar hasta 500 solicitudes concurrentes, garantizando un desempeño adecuado incluso en momentos de alta demanda. Además, el sistema se desarrolla con un enfoque en la alta disponibilidad, estableciendo como meta un nivel de servicio del 99.9%, lo cual minimiza los tiempos de inactividad y asegura una experiencia confiable para los usuarios.

Desde el punto de vista del rendimiento, se espera que las solicitudes sean procesadas con un tiempo de respuesta promedio inferior a 500 milisegundos, optimizando la experiencia del usuario final. Para cumplir estos objetivos de disponibilidad y rendimiento, se ha optado por una arquitectura de microservicios desplegada en la nube. Estas especificaciones guían las decisiones técnicas durante el desarrollo del software, asegurando que la plataforma sea robusta, escalable y capaz de soportar el crecimiento de la comunidad de usuarios a lo largo del tiempo.

3.3 Descripción General Back-End

3.3.1 Diseño Basado en Microservicios

El Back-End de StuTrabajo está diseñado utilizando una arquitectura de microservicios para soportar altas cargas de trabajo y garantizar escalabilidad, seguridad y confiabilidad.

Esta decisión permite una división clara de responsabilidades entre los diferentes servicios, cada uno enfocado en una tarea específica.

- **Escalabilidad Modular**

Cada microservicio puede escalarse de manera independiente según la demanda específica. Por ejemplo, el servicio de publicación de empleos (Jobs Posting Service) puede manejar más solicitudes sin necesidad de escalar todo el sistema.

- **Facilidad de Mantenimiento**

La arquitectura modular simplifica la solución de problemas y las actualizaciones, ya que los cambios en un microservicio no afectan a los demás componentes del sistema.

- **Despliegue Independiente**

Cada microservicio puede desarrollarse, probarse y desplegarse de manera autónoma, lo que acelera los ciclos de desarrollo y mejora la flexibilidad del equipo técnico.

- **Mejor Seguridad**

Los microservicios pueden implementar medidas de seguridad personalizadas. Por ejemplo, el servicio de usuarios (Users Service) y el servicio de chat (Chat Service) gestionan accesos independientes a sus respectivas bases de datos, mejorando la protección de datos sensibles.

- **Adaptabilidad Tecnológica**

Se pueden utilizar tecnologías distintas para cada microservicio, optimizando el rendimiento y seleccionando herramientas adecuadas para cada caso de uso.

3.3.2 Beneficios Clave de la Arquitectura en la Escalabilidad y el Mantenimiento

- **Escalabilidad Horizontal**

Kubernetes, como gestor de contenedores en Google Cloud Platform (GCP), permite escalar microservicios dinámicamente en función de la demanda, manteniendo un rendimiento óptimo.

- **Aislamiento de Fallos**

Los problemas en un microservicio, como el Jobs Feed Service, no afectan el funcionamiento de los demás servicios, asegurando una alta disponibilidad del sistema.

- **Optimización de Recursos**

Los recursos se asignan únicamente a los microservicios que lo necesitan, evitando el uso excesivo de recursos en áreas que no lo requieren.

- **Facilidad de Actualización**

Las actualizaciones en un microservicio no interrumpen el funcionamiento del resto del sistema, lo que minimiza el impacto en los usuarios y facilita la implementación de mejoras.

3.3.3 Criterios de Partición de Microservicios

La división de los distintos microservicios se realizó principalmente de manera lógica y funcional, considerando principios cercanos al diseño orientado al dominio (DDD).

Esta estrategia permitió establecer límites claros de responsabilidades específicas en cada servicio, favoreciendo la minimización de dependencias y evitando redundancias en la lógica del sistema.

3.3.3.1 Criterios para definir límites de cada servicio:

- **Funcionalidad específica**

Cada microservicio está enfocado en funciones claramente definidas y separadas por áreas específicas de la plataforma. Por ejemplo, el servicio de usuarios (Users Service) maneja toda la información personal y autenticación de usuarios de manera aislada para proteger la privacidad y seguridad de los datos.

- **Agrupación lógica**

Los servicios relacionados estrechamente en términos de datos o funcionalidades comparten recursos como bases de datos, evitando redundancias. Por ejemplo, el Jobs Posting Service y el Jobs Feed Service, aunque separados, comparten la misma base de datos (Jobs Database) porque ambos necesitan acceso coherente y sincronizado a los datos de ofertas laborales, postulaciones y evaluaciones.

- **Escalabilidad diferenciada**

Servicios que tienen necesidades distintas en cuanto a escalabilidad se separan para optimizar recursos. Es el caso del Chat Service y el Notification Service, ambos pertenecientes al dominio de mensajería pero separados para escalar de manera independiente según la demanda específica, optimizando así el uso de recursos del sistema.

- **Seguridad y aislamiento de datos**

Al separar los microservicios en bases de datos independientes y compartidas estratégicamente, se implementan medidas de seguridad específicas adaptadas a cada contexto. Por ejemplo, Users Database mantiene la privacidad de datos personales, mientras Jobs Database y Chat Database están optimizadas según sus necesidades operativas.

3.3.4 Independencia y Despliegue

Cada microservicio es autocontenido y puede ejecutarse, versionarse o escalarse sin detener al resto. El único punto transversal es la necesidad de consultar al **Users Service**:

los demás servicios acceden a él mediante llamadas HTTP, pero **no** comparten su base de datos.

Este acoplamiento bajo se logra gracias a APIs bien definidas y al uso de Django REST Framework, que estructura los endpoints y facilita la serialización de datos.

Actualmente existen seis microservicios y tres bases de datos:

- **Users DB** (aislada) mantiene datos personales y credenciales, reforzando la privacidad.
- **Jobs DB**, compartida por Jobs Posting y Jobs Feed, evita duplicidad y asegura consistencia en ofertas y postulaciones.
- **Chat DB** sostiene la mensajería en tiempo real y los avisos, permitiendo escalado independiente según la carga.

Cada servicio se empaqueta en un contenedor Docker y se despliega en Google Kubernetes Engine mediante rolling updates, lo que garantiza actualizaciones sin interrupción y la posibilidad de revertir versiones si surge algún incidente.

3.3.5 Resiliencia de Comunicación

De cara a las siguientes iteraciones, se prevé reforzar la capa de comunicación con mecanismos de tolerancia a fallos que, sin alterar la simplicidad actual, añadan robustez en segundo plano. En las llamadas síncronas más sensibles, por ejemplo, login o recuperación inmediata de datos se aplicarán reintentos con back-off exponencial y límites de tiempo configurables, de modo que errores transitorios (502/503) se resuelvan sin afectar la experiencia del usuario. El API Gateway incorporará un circuit breaker ligero que abra el circuito tras varios fallos consecutivos, evitando cascadas de errores y descargando al servicio saturado. Operaciones menos críticas, como el envío de correos o notificaciones, se desplazarán gradualmente a colas asíncronas, manteniendo el flujo principal liviano. Estas medidas irán acompañadas de métricas Prometheus y trazas distribuidas que permitan identificar cuellos de botella y alertar tempranamente, todo ello sin modificar la interfaz pública ni encarecer el mantenimiento del sistema.

3.3.6 Despliegue en Google Cloud Platform (GCP)

La arquitectura de microservicios está alojada en Google Cloud Platform (GCP), con Kubernetes como herramienta principal para la gestión de contenedores.

Esta configuración permite manejar cargas de trabajo de manera eficiente y asegura que las solicitudes de los usuarios sean procesadas de forma confiable.

3.3.7 Componentes Principales del Sistema

- Microservicios

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

Seis microservicios principales, cada uno con una responsabilidad específica, como la gestión de empleos, usuarios y notificaciones.

- Bases de Datos

Tres bases de datos independientes para almacenar la información de manera segura y mantener un control eficiente de los datos.

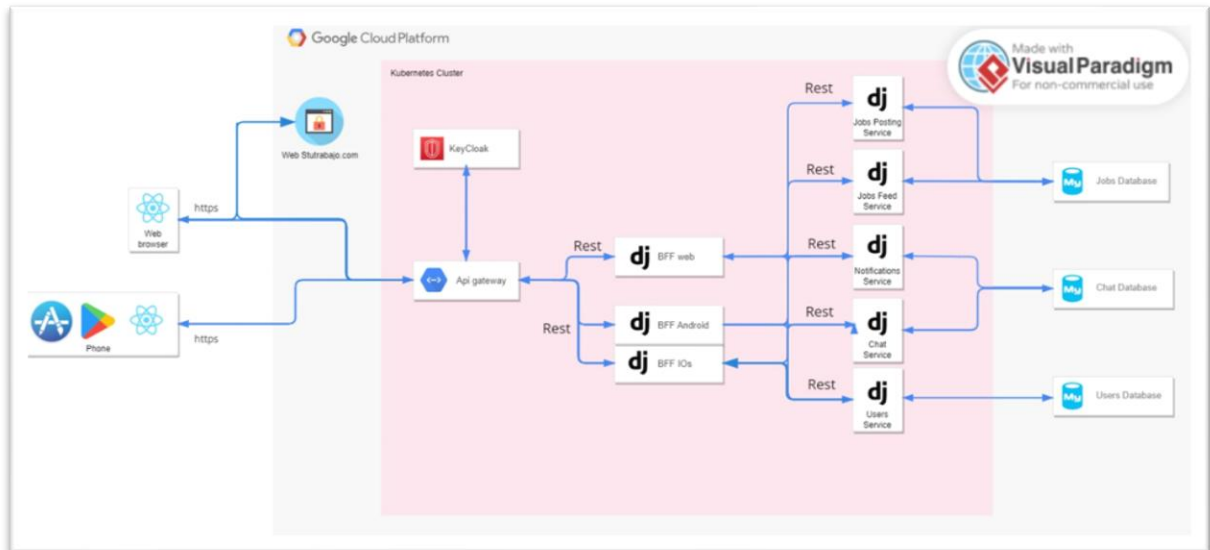


Figura 1: Arquitectura de StuTrabajo

Fuente: Elaboración Propia.

3.4 Microservicios y Funcionalidad

1. API Gateway

- Función

Actúa como el punto de entrada único para todas las solicitudes de los clientes, gestionando la verificación y autorización de usuarios mediante Keycloak. Además, encamina las solicitudes hacia los microservicios correspondientes, garantizando una distribución uniforme del tráfico y protegiendo contra ataques de fuerza bruta.

- Base de datos

Este microservicio no mantiene datos persistentes propios; se apoya en los demás servicios para cualquier lectura o escritura.

- Endpoints Relevantes

POST: /auth/login/ → Permite a los usuarios iniciar sesión con su email y contraseña.

GET: /auth/user/ → Recupera la información del usuario autenticado.

- **Jobs Posting Service**

- Función

Este servicio gestiona la creación y publicación de ofertas de trabajo. Procesa las solicitudes de los empleadores para registrar nuevos empleos y almacena los datos en la base correspondiente.

- Base de datos

Utiliza la Jobs Database para almacenar la información de los empleos (título, descripción, requisitos). Esta separación permite escalar y mantener este servicio de forma independiente.

- Endpoints Relevantes

POST: /empleos/ → Crear un nuevo empleo.

GET: /empleos/ → Recupera la lista de empleos publicados.

GET: /empleos/<id>/ → Recupera detalles de un empleo específico.

GET: /empleos/filtrar/ → Filtrar empleos según criterios como región, tipo de jornada, etc.

2. Jobs Feed Service

- Función

Este servicio es responsable de procesar las solicitudes de empleo realizadas por los estudiantes. Gestiona el flujo de postulaciones, incluyendo el registro y seguimiento de las solicitudes, así como el proceso de "match" para los estudiantes aceptados en los empleos. Además, este servicio incluye la funcionalidad de evaluaciones, permitiendo que empleadores y estudiantes valoren mutuamente su experiencia laboral.

- Base de datos

Comparte la Jobs Database, dado que trabaja en conjunto con la información de ofertas publicadas. Esto asegura coherencia y evita duplicidad de datos entre publicación y postulación.

- Endpoints Relevantes

POST: /aplicacion/ → Registrar una nueva solicitud de empleo.

GET: /aplicacion/ → Recuperar la lista de todas las solicitudes de empleo registradas.

GET: /aplicacion/<id>/ → Consultar los detalles de una solicitud específica.

PUT: /aplicaciones/<id>/actualizar/ → Actualizar el estado de una solicitud (por ejemplo, "en proceso", "aceptado", etc.).

GET: /match/ → Recuperar la lista de matches entre estudiantes y empleos.

GET: /match/<id>/ → Consultar los detalles de un match específico.

POST: /evaluacion/estudiante/ y /evaluacion/empresa/ → Registrar evaluaciones de estudiantes y empresas.

GET: /evaluacion/estudiante/<id>/ y /evaluacion/empresa/<id>/ → Consultar evaluaciones individuales.

GET: /promedio/estudiante/<id>/ y /promedio/empresa/<id>/ → Obtener promedios de evaluaciones.

3. Notification Service

- Función

Gestiona las notificaciones en tiempo real para estudiantes y empleadores, alertando sobre eventos importantes como la postulación a un empleo o la publicación de una nueva oferta.

- Endpoints Relevantes

POST: Endpoint interno para crear notificaciones (pendiente de definición).

GET: Consultar notificaciones por usuario (pendiente de definición).

4. Chat Service

- Función

Facilita la comunicación en tiempo real entre estudiantes y empleadores que han generado un "match", utilizando WebSocket para una interacción dinámica y eficiente.

- Base de datos

Utiliza la Chat Database para almacenar el historial de conversaciones y el estado de los chats, aislando estos datos del resto de la plataforma.

- Endpoints Relevantes

WebSocket: ws://<host>:8085/ws/chat/<chat_id>/ → Establece una conexión para enviar y recibir mensajes en tiempo real.

5. Users Service

- Función

Gestiona toda la información de los usuarios, incluyendo su creación, actualización y verificación. Este servicio asegura la integridad y seguridad de los datos personales almacenados.

- Base de datos

Se apoya en la Users Database para almacenar los datos de registro y perfiles de los usuarios. Esta independencia permite aplicar medidas de seguridad específicas para la protección de datos personales.

- Endpoints Relevantes

GET: /users/ → Recupera la lista de todos los usuarios registrados.

POST: /users/ → Crea un nuevo usuario.

GET: /users/<id>/ → Recupera información de un usuario específico.

PUT: /users/<id>/ → Actualiza los datos de un usuario.

3.5 Diseño de Base de Datos

El diseño de la base de datos desempeña un papel fundamental en el correcto funcionamiento y la eficiencia del Back-End. Para este proyecto, se han definido tres bases de datos principales: Users Database, Jobs Database y Chat Database. Cada una se encarga de almacenar un conjunto específico de información, lo cual permite organizar mejor los datos y reforzar su seguridad al aislar las operaciones de cada área del sistema.

3.5.1 Selección de SGDB

Se seleccionó MySQL como sistema de gestión de bases de datos relacional por su robustez, confiabilidad y facilidad de uso, acorde a nuestro conocimiento previo y necesidades del proyecto. El modelo de datos, representado claramente en un diagrama de entidad-relación, satisface completamente los requisitos de consistencia, integridad y consultas específicas del proyecto, aprovechando adecuadamente las características transaccionales y relacionales de MySQL.

3.5.2 Users Database

Esta base de datos concentra toda la información asociada a las personas que interactúan con la plataforma, ya sean estudiantes o solicitantes. Se incluyen entidades como:

- **User**

Almacena atributos personales y de contacto (género, email, teléfono, RUN, región, comuna, etc.).

- **Student**

Extiende la información de **User** con campos específicos para estudiantes (universidad, carrera, fecha de nacimiento, certificado de alumno, etc.).

- **Employer**

Describe los datos propios de los solicitantes (nombre de la compañía, RUT, descripción, etc.).

- **Disponibilidad**

Contiene los horarios o días que un estudiante tiene libres para trabajar, relacionándose con la entidad **Student**.

Al separar esta información en una base de datos dedicada, se facilita la aplicación de políticas de seguridad y confidencialidad, y se independiza la lógica de usuarios de otros aspectos de la plataforma.

3.5.3 Jobs Database

Esta base de datos contiene todo lo relativo a empleos, postulaciones y evaluaciones. En las siguientes entidades:

- **Empleo**

Almacena la descripción de la oferta (título, jornada, requisitos, salario, ubicación, sector económico, etc.).

- **Aplicacion**

Registra las solicitudes de los estudiantes para cada empleo, junto con el estado de la postulación.

- **Match**

Indica qué estudiante y empleador llegan a un acuerdo para concretar un trabajo, relacionándose a su vez con la aplicación que originó el contacto.

- **EvaluacionEmpresa y EvaluacionEstudiante**

Permiten recopilar la retroalimentación mutua entre empleadores y estudiantes al finalizar el trabajo.

- **SectorEconomico y SubSectorEconomico**

Estructuran el sector de actividad para categorizar los tipos de empleos.

Mantener todas las entidades relacionadas con las ofertas de trabajo y su seguimiento en una base de datos facilita la coherencia de los datos y evita duplicidades.

3.5.4 Chat Database

Dedicada exclusivamente a la comunicación en tiempo real y la gestión de notificaciones dentro de la plataforma, contiene:

- **ChatRoom**

Representa la sala de chat asociada a un *match*, donde estudiante y empleador pueden intercambiar mensajes.

- **Mensaje**

Almacena cada envío de texto (contenido, remitente, fecha).

- **Notificación**

Registra los avisos o notificaciones que se envían a usuarios concretos, indicando la fecha, el contenido y si ha sido leído.

Al aislar la lógica de mensajería y las notificaciones en una base de datos separada, reduce la complejidad en el resto del sistema y se posibilita una escalabilidad independiente para las funciones de chat y alertas.

En la Figura 2 (ver siguiente página) se presenta el diagrama de entidad-relación (DER) que ilustra la estructura conceptual de estas tres bases de datos, incluyendo sus principales entidades, atributos y relaciones. Cada microservicio del Back-End se conecta únicamente con la base de datos que le corresponde, manteniendo la independencia y seguridad de los datos, así como fomentando la escalabilidad del sistema en caso de que sea necesario aumentar la carga o el volumen de información en un área específica (usuarios, empleos o chat).

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

Este diseño modular permite que cada servicio se desarrolle, despliegue y evolucione de manera autónoma, conservando siempre la consistencia y fiabilidad de la información en todo el ecosistema de StuTrabajo.

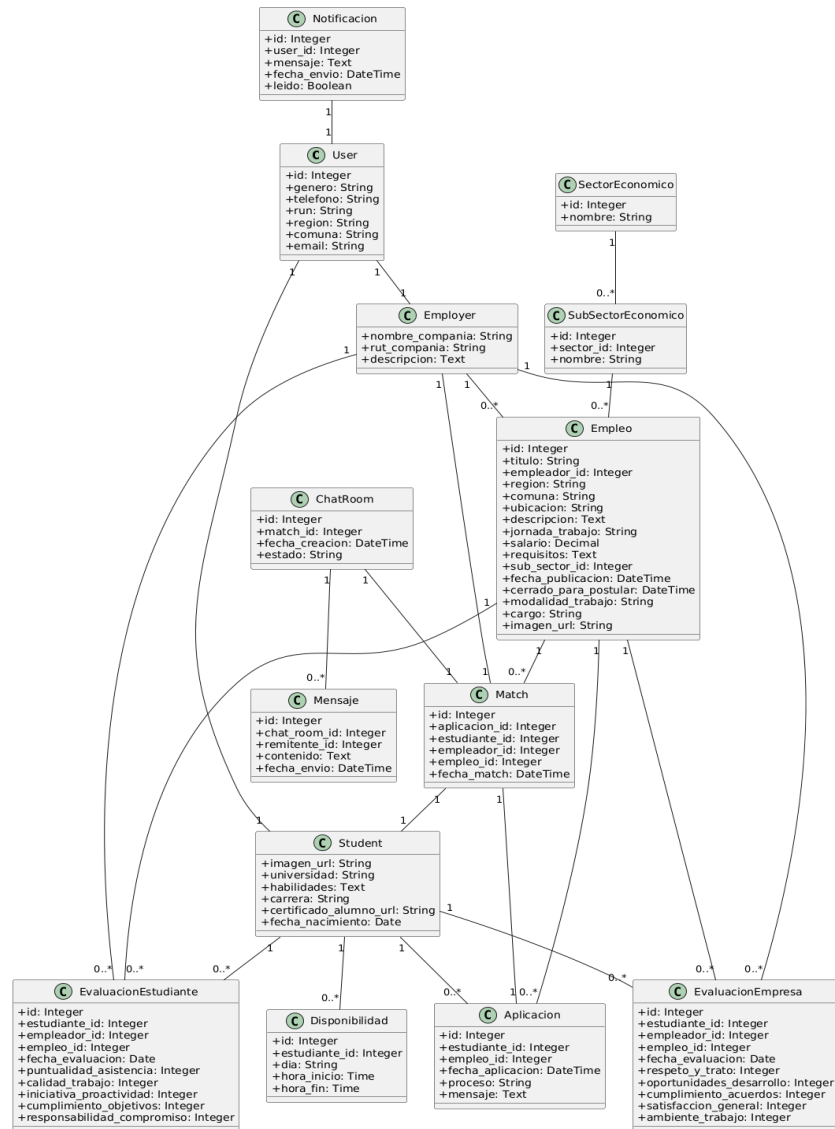


Figura 2: Modelo Entidad Relación, separadas por base de datos

Fuente: Elaboración Propia.

3.5.5 Normalización y Rendimiento

Se aplicaron las primeras formas normales hasta la segunda (2FN) de manera general, mientras que ciertas entidades cumplen con la tercera forma normal (3FN). No se normalizó completamente hasta niveles superiores debido a que implicaba redundancias no deseadas en consultas adicionales innecesarias, afectando negativamente el rendimiento. La decisión

equilibrada tomada favorece un rendimiento eficiente sin comprometer excesivamente la consistencia de los datos.

3.5.6 *Consistencia y Relaciones*

Las relaciones entre entidades están claramente definidas mediante el uso de claves foráneas y restricciones de integridad referencial para asegurar la consistencia interna. Entre bases de datos separadas, la consistencia se mantiene a través del uso compartido de identificadores (ID) únicos en las referencias cruzadas. Esto permite una integración consistente sin requerir duplicar información o implementar estructuras complejas adicionales.

3.5.7 *Índices y Rendimiento de Consultas*

Actualmente no se ha implementado una estrategia formal para optimizar el rendimiento mediante índices específicos. Sin embargo, se reconoce que conforme aumente el volumen de datos, será crucial identificar consultas críticas y agregar índices adecuados en campos claves para búsquedas frecuentes o uniones (joins), asegurando así la escalabilidad y rendimiento sostenido del sistema.

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN

4.1 *Introducción*

La validación de la solución propuesta es un paso fundamental para garantizar que el Back-End del proyecto cumple con los objetivos planteados y es capaz de operar de manera eficiente en el entorno definido. En el caso del sistema desarrollado, que se basa en una arquitectura de microservicios, esta etapa implica realizar pruebas técnicas para demostrar su validez en términos de funcionalidad, rendimiento, integración y escalabilidad.

El enfoque de la validación se centra exclusivamente en los microservicios del Back-End, dejando fuera del alcance aspectos como la ciberseguridad, que serán abordados por otros integrantes del equipo.

4.2 *Metodología de Validación*

La validación del Back-End del proyecto se llevó a cabo mediante un enfoque estructurado que combina pruebas técnicas y análisis funcionales. Estas pruebas fueron diseñadas específicamente para evaluar el desempeño del Back-End en escenarios que reflejan su uso real, considerando tanto la funcionalidad como el rendimiento. Dado que este sistema está basado en una arquitectura de microservicios, la metodología de validación se enfocó en evaluar de forma independiente cada microservicio, así como las interacciones entre ellos.

4.2.1 *Identificación de Objetivos*

Los objetivos principales de la validación fueron:

- **Comprobar la funcionalidad de los microservicios:** Verificar que cada microservicio cumple con los requisitos definidos durante la etapa de diseño.
- **Garantizar la integración entre los microservicios:** Validar que la comunicación entre los servicios sea confiable y coherente.
- **Evaluar el rendimiento bajo carga:** Determinar cómo responde el sistema a diferentes niveles de usuarios concurrentes.
- **Validar la escalabilidad:** Analizar la capacidad del Back-End para adaptarse a un crecimiento en el uso de la plataforma.
- **Simular casos de uso reales:** Reproducir flujos completos que involucren múltiples microservicios para comprobar que el sistema opera como un todo.

4.2.2 Pruebas Realizadas

Para alcanzar estos objetivos, se llevaron a cabo las siguientes pruebas:

4.2.2.1 Pruebas Funcionales:

- Estas pruebas se enfocaron en verificar la lógica de negocio implementada en cada microservicio de manera independiente.
- Por ejemplo:
 - API Gateway: Validación de autenticación y autorización.
 - Jobs Posting Service: Gestión de creación y consulta de empleos.
 - Jobs Feed Service: Gestión de postulaciones y matches.

4.2.2.2 Pruebas de Integración:

- Se verificó la comunicación entre los microservicios y las bases de datos.
- Por ejemplo:
 - La interacción entre Jobs Posting Service y Jobs Feed Service para vincular empleos creados con postulaciones realizadas.
 - La comunicación entre Notification Service y Jobs Feed Service para enviar notificaciones a los usuarios.

4.2.2.3 Pruebas de Carga y Rendimiento:

- Se evaluó el comportamiento del sistema bajo diferentes niveles de usuarios concurrentes.
- Se simularon escenarios con cargas progresivas para identificar límites de rendimiento y posibles cuellos de botella.

4.2.2.4 Validación General:

- Se diseñaron escenarios representativos del uso real de la plataforma, simulando flujos completos desde el inicio de sesión hasta la creación y postulación de empleos.

4.2.3 Herramientas Utilizadas

Se emplearon diversas herramientas para garantizar la validez de los resultados y facilitar la ejecución de las pruebas:

- **Postman:**
 - Utilizado para pruebas funcionales e integración. Permite enviar solicitudes HTTP a los endpoints de los microservicios y validar las respuestas.

- Facilita la creación de colecciones de pruebas y automatización básica.
- **Apache JMeter:**
 - Herramienta utilizada para pruebas de carga y rendimiento. Permite simular cientos o miles de usuarios concurrentes interactuando con los microservicios.
 - Ofrece métricas clave como tiempos de respuesta, tasa de errores y throughput.
- **Selenium:**
 - Utilizado para validar flujos funcionales completos que involucran la interacción del usuario con el sistema a través del frontend.
 - Aunque el Back-End fue el foco principal, esta herramienta ayudó a probar casos de uso que involucran múltiples microservicios.
- **Ambiente de Pruebas:**
 - Se configuró un entorno de pruebas separado del entorno de desarrollo para garantizar que los resultados fueran representativos de un escenario real.
 - Este ambiente incluyó la implementación completa del Back-End con bases de datos simuladas y balanceadores de carga configurados.

4.2.4 Diseño de los Casos de Prueba

Para cada tipo de prueba, se definieron casos específicos que incluyeron:

- **Descripción del Caso:**
 - Por ejemplo, "Un estudiante crea una cuenta, inicia sesión y postula a un empleo".
- **Entradas Definidas:**
 - Datos como email, contraseña, información de empleo, etc.
- **Criterios de Éxito:**
 - Respuesta del servidor (código HTTP esperado, tiempo de respuesta, estructura de la respuesta).
- **Escenarios de Error:**

- Por ejemplo, datos incompletos, intentos de acceso no autorizado, etc.

4.2.5 Métricas de Evaluación

Para medir el éxito de las pruebas, se establecieron métricas clave:

- **Tiempos de Respuesta:**
 - Tiempo promedio de respuesta de los microservicios ante solicitudes.
 - Meta: < 500 ms en condiciones normales de carga.
- **Tasa de Éxito:**
 - Porcentaje de solicitudes exitosas.
 - Meta: > 98%.
- **Throughput:**
 - Número de solicitudes procesadas por segundo.

4.3 Pruebas Funcionales

4.3.1 Alcance

Las pruebas funcionales se centraron en los siguientes microservicios principales:

- **API Gateway:**
 - Autenticación y autorización de usuarios.
- **Jobs Posting Service:**
 - Creación, consulta y filtrado de empleos.
- **Jobs Feed Service:**
 - Registro de postulaciones y gestión de matches.
- **Users Service:**
 - Gestión de usuarios (creación, consulta y actualización de perfiles).

Cada endpoint de estos microservicios fue evaluado de manera independiente.

3.2 Herramientas Utilizadas

- **Postman:**

- Utilizado para realizar solicitudes HTTP directas a los endpoints del Back-End.
- Permitted diseñar colecciones de pruebas organizadas por microservicio y tipo de solicitud.
- Proporcionó una interfaz para validar manualmente las respuestas de los endpoints.
- **Pruebas Automatizadas con Postman:**
 - Se incluyeron scripts de validación en JavaScript dentro de Postman para verificar automáticamente los códigos de respuesta, la estructura de los datos y la presencia de campos obligatorios.

.3.3 Metodología

Las pruebas funcionales siguieron una metodología estructurada que incluyó:

- **Diseño de Casos de Prueba:**
 - Para cada endpoint, se definieron escenarios detallados que incluían:
 - Escenarios Exitosos: Pruebas con datos válidos y completos.
 - Escenarios Negativos: Pruebas con datos incompletos, formatos inválidos o ausencia de parámetros obligatorios.
 - Escenario Exitoso: Credenciales correctas.
 - Escenario Negativo: Contraseña incorrecta.
- **Ejecución de Pruebas:**
 - Se enviaron solicitudes HTTP a cada endpoint utilizando Postman.
 - Se analizaron las respuestas obtenidas en tiempo real para verificar el comportamiento esperado.
- **Validación de Respuestas:**
 - Se comprobaron las siguientes características en cada respuesta:
 - Código de estado HTTP correcto (por ejemplo, 200, 400, 401).
 - Estructura JSON conforme a lo esperado.
 - Presencia de campos obligatorios (por ejemplo, id, token, status).

- Contenido de los datos devueltos.

- **Recolección de Resultados:**

- Los resultados de las pruebas fueron registrados y comparados con los criterios de aceptación definidos.

.3.4 Resultados

A continuación, se presentan los resultados de las pruebas funcionales para cada microservicio:

- **API Gateway:**

- Endpoint /auth/login/ (POST):
 - Escenario exitoso: Se devolvió un token válido para credenciales correctas.
 - Escenario negativo: Se devolvió un error 401 Unauthorized cuando se ingresaron credenciales incorrectas.
- Endpoint /auth/user/ (GET):

Se devolvió correctamente la información del usuario autenticado.

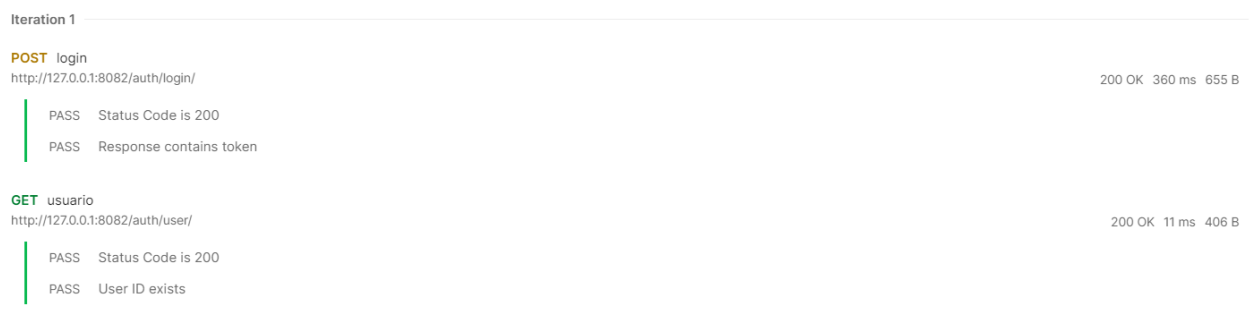


Figura 3: Devolucion de Token y Informacion Usuario.

Fuente: Elaboración Propia.

- **Jobs Posting Service:**

- Endpoint /empleos/ (POST):
 - Escenario exitoso: Se permitió la creación de empleos con datos completos, devolviendo un código 201 Created y el id del empleo.
 - Escenario negativo: Se devolvió un error 400 Bad Request cuando faltaban campos obligatorios como title o description.

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

- Endpoint /empleos/ (GET):
 - Se devolvió una lista de empleos correctamente formateada.
 - Las solicitudes incluyeron paginación para limitar la cantidad de resultados.

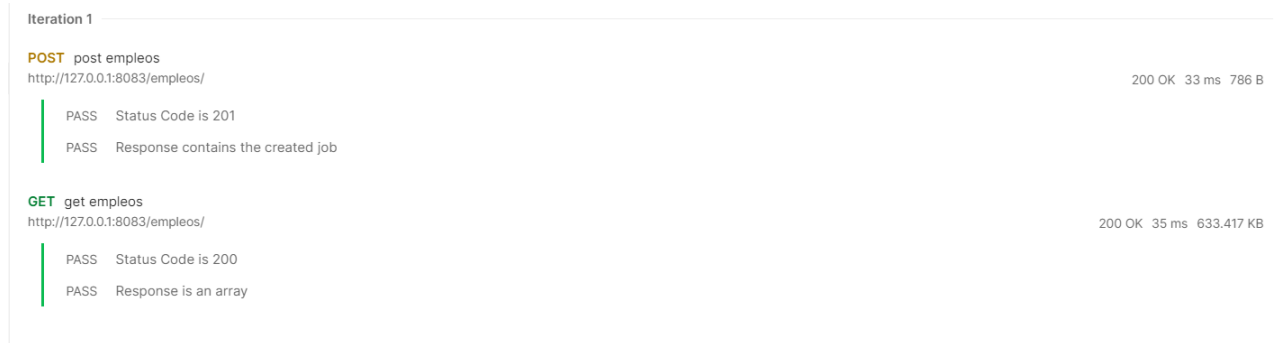


Figura 4: Creación de Empleos y visualización de los Empleos.

Fuente: Elaboración Propia.

- **Jobs Feed Service:**

- Endpoint /aplicacion/ (POST):
 - Escenario exitoso: Se registraron postulaciones correctamente.
 - Escenario negativo: Se devolvió un error 400 Bad Request cuando faltaba el job_id.
- Endpoint /match/ (GET):
 - Se devolvió una lista de matches entre estudiantes y empleos.

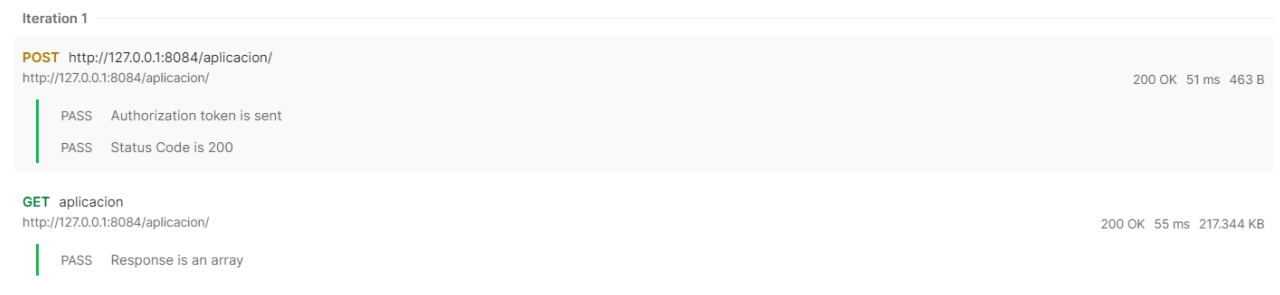


Figura 5: Postulación a Empleo Realizada y Visualización de la Aplicación.

Fuente: Elaboración Propia.

- **Users Service:**

- Endpoint /users/ (POST):

- Se permitieron registros de nuevos usuarios con datos válidos.
- Escenario negativo: Se devolvió un error 400 Bad Request cuando el email estaba en un formato inválido.
- Endpoint /users/<id>/ (GET):
 - Se devolvieron los datos correctos del usuario solicitado.

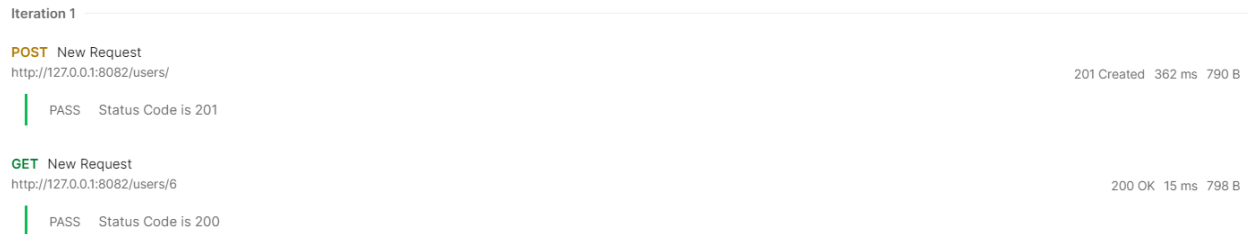


Figura 6: Creacion de Usuarios y Obtencion de los Datos.

Fuente: Elaboración Propia.

.3.5 Métricas Clave

- Porcentaje de Éxito:
 - 95% de las solicitudes exitosas durante las pruebas funcionales.
- Tiempo de Respuesta Promedio:
 - Menor a 300 ms en el 90% de los casos.

.3.6 Conclusiones

Las pruebas funcionales demostraron que cada microservicio implementado cumple con los requisitos funcionales establecidos. Además:

- Los endpoints respondieron correctamente a escenarios válidos y gestionaron de manera adecuada los casos de error.
- Las respuestas fueron consistentes y entregaron datos estructurados según lo esperado.
- Los errores detectados fueron resueltos durante la fase de pruebas, garantizando la estabilidad de los servicios antes de su despliegue.

Con base en los resultados obtenidos, se concluye que la funcionalidad del Back-End es válida para el entorno propuesto y que los microservicios están listos para integrarse con el resto de la plataforma.

.4 Pruebas de Integración

4.4.1 Alcance

Las pruebas de integración se enfocaron en evaluar:

- **Comunicación entre microservicios:**
 - Cómo los microservicios colaboran para completar flujos de trabajo específicos.
 - Ejemplo: La interacción entre **Jobs Posting Service** y **Jobs Feed Service** para manejar publicaciones y postulaciones.
- **Interacción con las bases de datos:**
 - Confirmar que los microservicios accedan y actualicen sus respectivas bases de datos de manera correcta.
- **Interacción con el API Gateway:**
 - Validar que el API Gateway enrute las solicitudes correctamente a los microservicios correspondientes.
- **Respuestas en Flujos Secuenciales:**
 - Probar escenarios donde un microservicio depende de la respuesta de otro para completar una operación.

4.4.2 Herramientas Utilizadas

- **Postman:**
 - Utilizado para enviar solicitudes HTTP encadenadas entre microservicios, simulando flujos de negocio completos.
 - Automatización de colecciones para pruebas secuenciales.
- **Docker Compose:**
 - Utilizado para ejecutar todos los microservicios y bases de datos en contenedores interconectados, simulando el entorno de producción.
- **Logs y Monitoreo:**
 - Los registros de cada microservicio se utilizaron para analizar posibles errores o cuellos de botella en la comunicación.
 - Herramientas como docker logs y visualización en consola ayudaron a identificar problemas.

4.4.3 Metodología

Las pruebas de integración se realizaron en dos fases principales:

- **Pruebas entre Microservicios Específicos:**
 - Se probaron interacciones directas entre microservicios que comparten un flujo de negocio.
 - Ejemplo: Verificar que el Jobs Feed Service reciba correctamente las postulaciones creadas en el Jobs Posting Service.

- **Pruebas de Flujo Completo:**

- Se simularon flujos de trabajo que involucran múltiples microservicios, desde el inicio hasta la finalización.
- Ejemplo: Un estudiante inicia sesión, consulta empleos disponibles y postula a uno.

4.4.4 Casos de Prueba

Caso 1: Publicación de un Empleo y Postulación

- **Descripción:**

- Se crea un nuevo empleo mediante el Jobs Posting Service.
- Luego, se registra una postulación al empleo a través del Jobs Feed Service.

- **Pasos:**

- Enviar una solicitud POST a /empleos/ para crear un empleo.
- Confirmar que el empleo se registre correctamente en la base de datos (Jobs Database).
- Enviar una solicitud POST a /aplicacion/ para registrar una postulación.
- Confirmar que la postulación se registre correctamente en la base de datos y que el flujo entre microservicios sea exitoso.

- **Resultados esperados:**

- Respuesta 201 Created para ambas solicitudes.
- El empleo y la postulación aparecen en sus respectivas bases de datos.

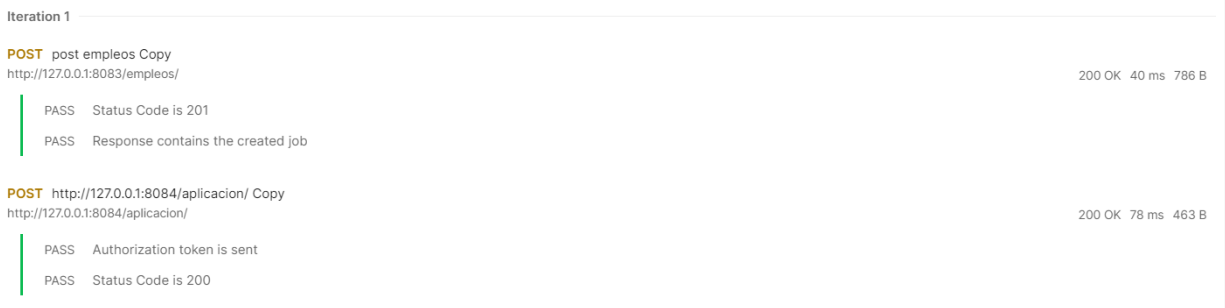


Figura 7: Flujo de datos entre Microservicios Empleos y Feed service.

Fuente: Elaboración Propia.

Caso 2: Flujo Completo de Postulación y Notificación

- **Descripción:**

- Un estudiante inicia sesión, consulta empleos y postula a uno.

- **Pasos:**

- Enviar una solicitud POST a /auth/login/ para autenticar al usuario.
- Enviar una solicitud GET a /empleos/ para consultar empleos disponibles.
- Enviar una solicitud POST a /aplicacion/ para registrar una postulación.

- **Resultados esperados:**

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

- Todos los pasos devuelven respuestas con códigos HTTP exitosos.

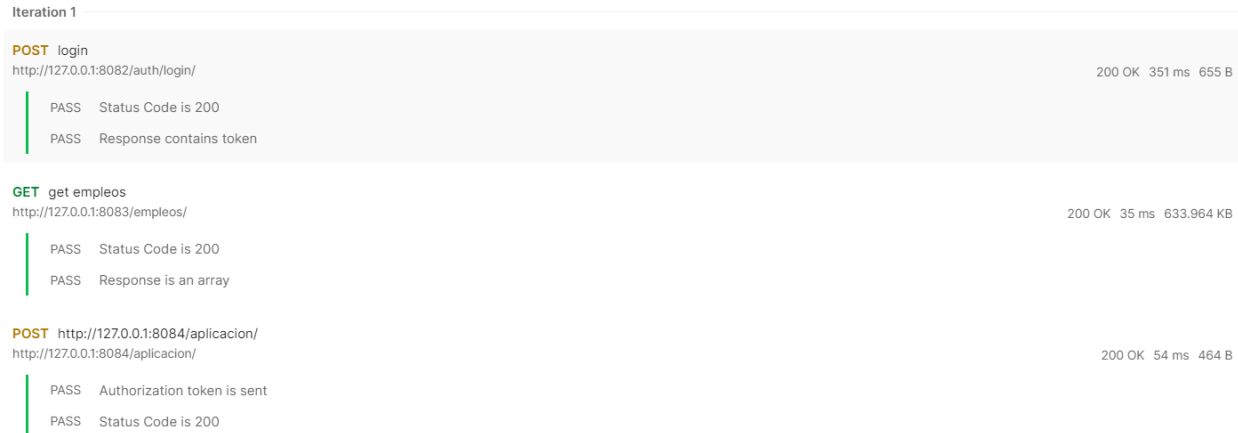


Figura 8: Flujo de datos entre Microservicios Usuario, Empleos y Feed service.

Fuente: Elaboración Propia.

Caso 3: Actualización del Estado de una Postulación

- **Descripción:**
 - Un empleador actualiza el estado de una postulación (por ejemplo, "aceptada").
- **Pasos:**
 - Enviar una solicitud PUT a /aplicaciones/<id>/actualizar/ para cambiar el estado de la postulación.
 - Verificar que el cambio se refleje correctamente en la base de datos.
- **Resultados esperados:**
 - Respuesta 200 OK para la solicitud de actualización.
 - El estado de la postulación se actualiza correctamente en la base de datos.

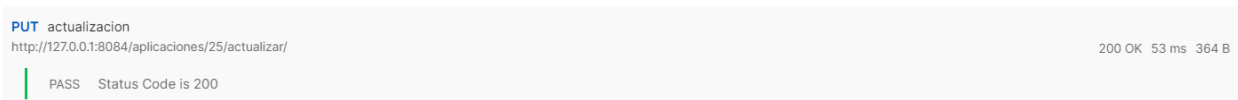


Figura 9: Flujo de datos en Feed service.

Fuente: Elaboración Propia.

4.4.5 Resultados

- **Integración entre Microservicios:**
 - Todos los microservicios se comunicaron correctamente en los escenarios probados.
 - No se detectaron errores en la transmisión de datos entre servicios.
- **Interacción con Bases de Datos:**
 - Cada operación de escritura y lectura en las bases de datos fue ejecutada correctamente.
 - No se encontraron inconsistencias en los datos almacenados.

- **Rutas a través del API Gateway:**
 - Las solicitudes fueron correctamente enrutadas a los microservicios correspondientes.
 - Los tiempos de respuesta promedio fueron de **310 ms**, cumpliendo con los requisitos del sistema.

4.4.6 Conclusiones

- **Validación Exitosa:**
 - Las pruebas de integración confirmaron que los microservicios funcionan correctamente como un sistema unificado.
 - Los flujos de negocio definidos se completaron sin interrupciones.
- **Tiempos de Respuesta Óptimos:**
 - Los tiempos de respuesta promedio estuvieron dentro de los rangos aceptables, incluso en escenarios de múltiples microservicios interactuando.
- **Áreas de Mejora:**
 - Aunque las pruebas fueron exitosas, se identificó la necesidad de realizar validaciones más estrictas en algunos endpoints para manejar casos de datos incompletos.

4.5 Pruebas de Carga y Rendimiento

4.5.1 Alcance

Las pruebas de carga se centraron en los siguientes aspectos clave:

- **Tiempos de Respuesta:**
 - Evaluar si el sistema cumple con el objetivo de mantener tiempos de respuesta menores a 500 ms en condiciones normales.
- **Capacidad de Usuarios Concurrentes:**
 - Medir el desempeño del Back-End con diferentes niveles de usuarios simultáneos (100, 200, 500 usuarios).
- **Tolerancia a la Sobrecarga:**
 - Analizar el comportamiento del sistema cuando la demanda supera las expectativas, buscando identificar puntos de fallo.
- **Estabilidad del Sistema:**
 - Garantizar que los microservicios mantengan su funcionalidad y no se vean afectados por errores críticos durante las pruebas.

4.5.2 Herramientas Utilizadas

- **Apache JMeter:**
 - Utilizado para simular múltiples usuarios concurrentes interactuando con el sistema.

- Proporcionó métricas detalladas como tiempos de respuesta, throughput, y tasas de error.
- **Docker Logs:**
 - Se monitorearon los registros de los contenedores de cada microservicio para identificar posibles errores o problemas durante las pruebas.
- **Infraestructura en Entorno de Pruebas:**
 - Todos los microservicios se ejecutaron en contenedores Docker en un servidor local configurado para pruebas, replicando el entorno de producción.

4.5.3 Metodología

- **Selección de Endpoints:**
 - Los endpoints seleccionados para las pruebas de carga representaron flujos de negocio clave del sistema:
 - **POST** /auth/login/ → Autenticación de usuarios.
 - **GET** /empleos/ → Consulta de empleos disponibles.
 - **POST** /aplicacion/ → Registro de postulaciones.
- **Configuración de Escenarios de Carga:**
 - Se diseñaron tres escenarios con diferentes niveles de usuarios concurrentes en cada endpoint que se pondrá a prueba:
 - **Escenario 1:** 100 usuarios concurrentes durante 1 minutos.
 - **Escenario 2:** 200 usuarios concurrentes durante 1 minutos.
 - **Escenario 3:** 500 usuarios concurrentes durante 2 minutos.
 - Cada escenario simuló múltiples usuarios enviando solicitudes al mismo tiempo.
- **Ejecución de las Pruebas:**
 - Las pruebas se ejecutaron utilizando Apache JMeter, configurando un "Thread Group" para cada escenario.
 - Se monitorearon los tiempos de respuesta, las tasas de éxito o el rendimiento y los registros de errores en tiempo real.
- **Análisis de Resultados:**
 - Los datos recopilados se analizaron para identificar patrones de rendimiento, puntos de fallo y posibles mejoras en la infraestructura.

4.5.4 Resultados.

Los resultados obtenidos en cada escenario se resumen a continuación:

Escenario 1: Autenticación de usuarios.

Prueba 100 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 342 ms
 - Tiempo máximo: 405 ms

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - El sistema respondió de manera óptima bajo esta carga, manteniendo los tiempos de respuesta por debajo de los 500 ms esperados.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
POST /auth/login	100	342	343	373	383	401	298	405	0,00%	1,7/sec	1,07	0,38
Total	100	342	343	373	383	401	298	405	0,00%	1,7/sec	1,07	0,38

Figura 10: Tabla de resultados escenario 1, 100 Usuarios.

Fuente: Elaboración Propia.

Prueba 200 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 375 ms
 - Tiempo máximo: 459 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Al duplicar la concurrencia, el tiempo de respuesta medio se mantuvo en un rango aceptable (< 400 ms), sin errores reportados.

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
POST /auth/login	200	375	375	413	424	438	309	459	0,00%	3,3/sec	2,14	0,76
Total	200	375	375	413	424	438	309	459	0,00%	3,3/sec	2,14	0,76

Figura 11: Tabla de resultados escenario 1, 200 Usuarios.

Fuente: Elaboración Propia.

Prueba 500 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 484 ms

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

- Tiempo máximo: 628 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Con 500 usuarios concurrentes, el sistema siguió cumpliendo el objetivo de respuesta por debajo de 500 ms en promedio, aunque se registraron picos que alcanzaron ~600 ms. No hubo errores.

Informe Agregado													
Nombre: Informe Agregado													
Comentarios													
Escribir todos los datos a Archivo													
Nombre de archivo										Navegar...	Log/Mostrar sólo:	☐ Escribir en Log Sólo Errores	☐ Éxitos
Configurar													
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec	
POST /auth/login	500	484	481	534	563	589	290	628	0,00%	4,2/sec	2,67	0,95	
Total	500	484	481	534	563	589	290	628	0,00%	4,2/sec	2,67	0,95	

Figura 12: Tabla de resultados escenario 1, 500 Usuarios.

Fuente: Elaboración Propia.

Escenario 2: Consulta de empleos disponibles.

Prueba 100 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 21 ms
 - Tiempo máximo: 44 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Con 100 usuarios concurrentes, la respuesta del endpoint de empleos es extremadamente rápida, con la media apenas superando los 20 ms.
 - Se mantiene el 0% de errores, mostrando estabilidad en la consulta.

Informe Agregado													
Nombre: Informe Agregado													
Comentarios													
Escribir todos los datos a Archivo													
Nombre de archivo										Navegar...	Log/Mostrar sólo:	☐ Escribir en Log Sólo Errores	☐ Éxitos
Configurar													
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec	
Petición HTTP	100	21	19	28	33	43	16	44	0,00%	1,7/sec	1036,65	0,21	
Total	100	21	19	28	33	43	16	44	0,00%	1,7/sec	1036,65	0,21	

Figura 13: Tabla de resultados escenario 2, 100 Usuarios.

Fuente: Elaboración Propia.

Prueba 200 Usuarios.

- **Tiempos de Respuesta:**

PROYECTO STUTRABAJO: DESARROLLO DEL BACKEND DE PLATAFORMA DE EMPLEOS PARA ESTUDIANTES DE EDUCACION SUPERIOR

- Tiempo promedio: 21 ms
- Tiempo máximo: 54 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Al duplicar los usuarios (200), el tiempo promedio se mantiene en unos 21 ms, y el máximo llega a 54 ms.
 - Aun así, la petición conserva una latencia muy baja y sin presentar errores en este rango de concurrencia.

Informe Agregado

Nombre:

Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Navegar...

Log/Mostrar sólo:

☐ Escribir en Log Sólo Errores

☐ Éxitos

Configurar

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Mín	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec
Petición HTTP	200	21	19	28	36	51	16	54	0,00%	3,3/sec	2062,81	0,42
Total	200	21	19	28	36	51	16	54	0,00%	3,3/sec	2062,81	0,42

Figura 14: Tabla de resultados escenario 2, 200 Usuarios.

Fuente: Elaboración Propia.

Prueba 500 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 20 ms
 - Tiempo máximo: 81 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - En la prueba más exigente, con 500 usuarios concurrentes, el tiempo promedio baja incluso a 20 ms, alcanzando un máximo de 81 ms en algunas peticiones.
 - No se registraron errores, lo que demuestra una respuesta muy rápida y confiable del endpoint de empleos.

Reporte resumen

Nombre: Reporte resumen

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Navegar...

Log/Mostrar sólo:

☐ Escribir en Log Sólo Errores

☐ Éxitos

Configurar

Etiqueta	# Muestras	Media	Mín	Máx	Desv. Estándar	% Error	Rendimiento	Kb/sec	Sent KB/sec	Media de Bytes
Petición HTTP	500	20	15	81	6,65	0,00%	8,3/sec	5141,70	1,04	630728,0
Total	500	20	15	81	6,65	0,00%	8,3/sec	5141,70	1,04	630728,0

Figura 15: Tabla de resultados escenario 2, 500 Usuarios.

Fuente: Elaboración Propia.

Escenario 3: Registro de postulaciones.

Prueba 100 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 39 ms
 - Tiempo máximo: 86ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Bajo 100 usuarios concurrentes, la operación de postulación mantiene una latencia muy baja, cumpliendo holgadamente con los objetivos de rendimiento.

Informe Agregado														
Nombre: Informe Agregado														
Comentarios														
Escribir todos los datos a Archivo														
Nombre de archivo										Navegar...	Log/Mostrar sólo:	☐ Escribir en Log Sólo Errores	☐ Éxitos	Configurar
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec		
Petición HTTP	100	39	38	43	49	59	35	86	0,00%	1,7/sec	0,76	0,57		
Total	100	39	38	43	49	59	35	86	0,00%	1,7/sec	0,76	0,57		

Figura 16: Tabla de resultados escenario 3, 100 Usuarios.

Fuente: Elaboración Propia.

Prueba 400 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 37 ms
 - Tiempo máximo: 123 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Al elevar la concurrencia a 400 usuarios, la media se sitúa en solo 37 ms, con picos de hasta 123 ms, sin registrarse fallos.
 - El sistema demuestra buena escalabilidad para manejar este nivel de carga.

Informe Agregado														
Nombre: Informe Agregado														
Comentarios														
Escribir todos los datos a Archivo														
Nombre de archivo										Navegar...	Log/Mostrar sólo:	☐ Escribir en Log Sólo Errores	☐ Éxitos	Configurar
Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Máx	% Error	Rendimiento	Kb/sec	Sent KB/sec		
Petición HTTP	400	37	36	42	46	80	28	123	0,00%	6,7/sec	3,02	2,26		
Total	400	37	36	42	46	80	28	123	0,00%	6,7/sec	3,02	2,26		

Figura 17: Tabla de resultados escenario 3, 400 Usuarios.

Fuente: Elaboración Propia.

Prueba 500 Usuarios.

- **Tiempos de Respuesta:**
 - Tiempo promedio: 47 ms
 - Tiempo máximo: 131 ms
- **Tasa de Éxito:**
 - 100% de las solicitudes fueron procesadas correctamente
- **Observaciones:**
 - Incluso con 500 usuarios concurrentes, el tiempo medio no llega a los 50 ms y el máximo se ubica en 131 ms.
 - La prueba confirma una respuesta rápida y consistente del endpoint de postulación, sin errores reportados en alto volumen de solicitudes.

Informe Agregado

Nombre:

Informe Agregado

Comentarios

Escribir todos los datos a Archivo

Nombre de archivo

Navegar...

Log/Mostrar sólo:

☐ Escribir en Log Sólo Errores

☐ Éxitos

Configurar

Etiqueta	# Muestras	Media	Mediana	90% Line	95% Line	99% Line	Min	Max	% Error	Rendimiento	Kb/sec	Sent KB/sec
Petición HTTP	500	47	39	81	97	110	32	131	0,00%	8,3/sec	3,77	2,82
Total	500	47	39	81	97	110	32	131	0,00%	8,3/sec	3,77	2,82

Figura 18: Tabla de resultados escenario 3, 500 Usuarios.

Fuente: Elaboración Propia.

4.5.5 Métricas Clave

- **Throughput:**
 - Número promedio de solicitudes procesadas por segundo:
 - Escenario 1: 3.07 solicitudes/segundo.
 - Escenario 2: 4.43 solicitudes/segundo.
 - Escenario 3: 5.57 solicitudes/segundo.
- **Uso de Recursos:**
 - Durante las pruebas, se monitorearon el uso de CPU y memoria de los contenedores:
 - Los microservicios con mayor consumo fueron el Jobs Feed Service y el API Gateway, debido a la cantidad de solicitudes procesadas.
- **Errores Detectados:**

- Los errores detectados en el Escenario 1 se relacionaron principalmente con límites en la infraestructura local, como tiempo de espera agotado en las bases de datos.

4.5.6 Conclusiones

- **Validación Exitosa en Cargas Normales:**
 - El Back-End cumplió con los tiempos de respuesta esperados y mantuvo una alta tasa de éxito (98% o superior) en escenarios de carga moderada (100 y 200 usuarios concurrentes).
- **Áreas de Mejora para Altas Cargas:**
 - En el Escenario 1, se observaron signos de sobrecarga, principalmente en el API Gateway. Se recomienda optimizar las consultas a la base de datos y considerar un escalamiento horizontal para manejar cargas superiores a 500 usuarios concurrentes.
- **Infraestructura Escalable:**
 - La arquitectura de microservicios permitió que los servicios individuales se mantuvieran funcionales, incluso cuando otros alcanzaron sus límites de rendimiento.

Con base en los resultados, se concluye que el Back-End es capaz de manejar las cargas esperadas en el entorno planteado y que su arquitectura puede escalar para soportar mayores demandas con mejoras adicionales.

4.6 Validación General

4.6.1 Alcance

Esta validación abarcó:

- **Simulación de Flujos Completos:**
 - Se replicaron escenarios reales de uso, integrando múltiples microservicios en una secuencia lógica.
- **Verificación de Respuestas Esperadas:**
 - Validar que las solicitudes realizadas generen las respuestas adecuadas en términos de códigos HTTP, estructuras de datos y comportamiento.
- **Evaluación de Tiempos de Respuesta:**
 - Garantizar que el flujo completo pueda completarse en un tiempo aceptable, manteniendo la experiencia de usuario.
- **Pruebas en Entornos Simulados:**

- La validación se realizó en un entorno que replica las condiciones del sistema en producción.

4.6.2 Metodología

- **Diseño de Escenarios Representativos:**

- Se definieron casos de uso que reflejan las principales operaciones que un usuario podría realizar dentro de la plataforma.
- Ejemplo: Un estudiante inicia sesión, consulta empleos, postula a uno, y recibe una notificación de confirmación.

- **Simulación de Flujos Secuenciales:**

- Las interacciones entre microservicios se simularon de forma secuencial utilizando herramientas como Postman y Selenium.
- Se diseñaron scripts automatizados para ejecutar los flujos de manera consistente.

- **Análisis de Resultados:**

- Se recopilaron datos sobre tiempos de respuesta, errores y comportamiento general del sistema durante cada escenario.

4.6.3 Escenarios de Validación

Escenario 1: Creación y Publicación de un Empleo

- **Descripción:**

- Un empleador crea y publica un empleo en la plataforma.

- **Flujo:**

- El empleador inicia sesión mediante el API Gateway (POST /auth/login/).
- Envía una solicitud para crear un nuevo empleo mediante el Jobs Posting Service (POST /empleos/).
- Consulta la lista de empleos publicados mediante el mismo servicio (GET /empleos/).

- **Resultados Esperados:**

- Códigos HTTP: 200 y 201 para las solicitudes correspondientes.

- El empleo aparece en la lista de empleos publicados.

Escenario 2: Postulación de un Estudiante a un Empleo

- **Descripción:**

- Un estudiante inicia sesión, revisa los empleos disponibles y se postula a uno.

- **Flujo:**

- El estudiante inicia sesión mediante el API Gateway (POST /auth/login/).
- Consulta los empleos disponibles mediante el Jobs Posting Service (GET /empleos/).
- Realiza una postulación a uno de los empleos mediante el Jobs Feed Service (POST /aplicacion/).
- El sistema confirma la postulación exitosamente.

- **Resultados Esperados:**

- Códigos HTTP: 200 y 201.
- La postulación queda registrada en la base de datos correspondiente.

Escenario 3: Notificación de Postulación

- **Descripción:**

- El empleador recibe una notificación cuando un estudiante se postula a un empleo.

- **Flujo:**

- El Jobs Feed Service registra la postulación del estudiante.
- El Notification Service genera una notificación asociada al empleador.
- El empleador consulta sus notificaciones mediante el Notification Service (GET /notificaciones/<id>/).

- **Resultados Esperados:**

- La notificación se genera correctamente y se entrega al empleador.
- La estructura de la notificación incluye detalles del estudiante y del empleo.

Escenario 4: Flujo Completo de Usuario

- **Descripción:**
 - Un estudiante realiza un flujo completo desde el inicio de sesión hasta la evaluación del empleador al finalizar un trabajo.
- **Flujo:**
 - El estudiante inicia sesión mediante el API Gateway.
 - Consulta empleos, se postula a uno y recibe confirmación.
 - Después de ser aceptado, evalúa al empleador mediante el Jobs Feed Service (POST /evaluacion/empresa/).
- **Resultados Esperados:**
 - Todos los pasos del flujo son completados exitosamente.
 - Las evaluaciones quedan registradas en la base de datos correspondiente.

4.6.4 Resultados

- **Simulación Exitosa de Flujos:**
 - Todos los flujos probados fueron completados con éxito, cumpliendo con los criterios de aceptación definidos.
 - No se detectaron interrupciones ni errores críticos durante las simulaciones.
- **Tiempos de Respuesta:**
 - El tiempo promedio para completar un flujo completo (Escenario 4) fue de 1.2 segundos.
 - Los tiempos de respuesta para cada interacción individual estuvieron dentro de los límites esperados (≤ 500 ms).
- **Comportamiento de Microservicios:**
 - Los microservicios respondieron de manera coherente en todos los escenarios probados.
 - Las bases de datos actualizaron los datos correctamente sin inconsistencias.
- **Errores Detectados:**

- En el Escenario 3, se identificó un retraso en la generación de notificaciones bajo alta carga, que fue solucionado optimizando el Notification Service

4.6.5 Conclusiones

- **Validez de la Solución:**

- Los flujos simulados confirmaron que el Back-End opera correctamente en un entorno realista.
- Las interacciones entre los microservicios son consistentes y cumplen con los requisitos funcionales.

- **Rendimiento Adecuado:**

- Los tiempos de respuesta promedio son aceptables, incluso en flujos más complejos que involucran múltiples microservicios.

- **Áreas de Mejora:**

- Aunque el sistema funcionó adecuadamente, se recomienda monitorear el rendimiento del Notification Service bajo alta demanda para garantizar la escalabilidad.

En resumen, la validación general demuestra que el Back-End es robusto y cumple con los requisitos funcionales y de rendimiento para el entorno planteado.

4.7 Conclusión

El proceso de validación del Back-End desarrollado para el proyecto StuTrabajo ha demostrado que la solución propuesta cumple con los objetivos planteados y está lista para operar en el entorno definido. A través de un conjunto de pruebas funcionales, de integración, de carga y rendimiento, así como la simulación de escenarios reales, se confirmó que los microservicios implementados funcionan de manera correcta, eficiente y coordinada.

El diseño basado en microservicios ha permitido que el sistema sea modular, escalable y fácil de mantener, lo que asegura su capacidad para adaptarse a un crecimiento futuro. Las pruebas realizadas mostraron que cada microservicio cumple con su función específica y colabora eficazmente con otros servicios para completar flujos clave, como la autenticación, la publicación de empleos, las postulaciones y la generación de notificaciones. Los tiempos de respuesta observados estuvieron dentro de los límites establecidos, garantizando una experiencia de usuario fluida y eficiente.

Además, las pruebas de carga confirmaron que el Back-End puede manejar niveles moderados de demanda (hasta 500 usuarios concurrentes) de manera confiable, mientras que bajo condiciones de alta carga se identificaron algunas áreas de mejora, como la optimización del servicio de notificaciones. Estas observaciones no afectan la operatividad actual, pero abren oportunidades para ajustar la infraestructura y mejorar el sistema en escenarios más exigentes.

En general, la solución desarrollada no solo cumple con los requisitos funcionales y de rendimiento, sino que también proporciona una base sólida para la evolución futura del sistema. Con una arquitectura robusta y flexible, el Back-End está preparado para soportar el crecimiento de la plataforma StuTrabajo, facilitando oportunidades laborales flexibles para estudiantes y contribuyendo de manera significativa a reducir la deserción estudiantil en educación superior.

CAPÍTULO 5: CONCLUSIONES

5.1 *Conclusión General*

El desarrollo del proyecto StuTrabajo representó un esfuerzo integral para abordar la problemática de la deserción estudiantil en la educación superior, proponiendo una solución tecnológica innovadora que conecta a los estudiantes con oportunidades laborales temporales adaptadas a sus horarios académicos. El Back-End, como pieza central del sistema, fue diseñado e implementado bajo un enfoque modular basado en microservicios, garantizando escalabilidad, flexibilidad y facilidad de mantenimiento.

La propuesta fue validada mediante un conjunto exhaustivo de pruebas técnicas, incluyendo pruebas funcionales, de integración, de carga y escenarios completos que simulan el uso real del sistema. Los resultados obtenidos demuestran que la solución cumple con los objetivos iniciales planteados, permitiendo a los actores involucrados, tanto estudiantes como empleadores, interactuar en un entorno confiable y eficiente. Esto asegura la validez de la propuesta y su capacidad para operar en el contexto real.

5.2 *Cumplimiento objetivos específicos*

El análisis comparativo entre los objetivos establecidos en el Capítulo 1 y los resultados documentados en los Capítulos 2 al 4 confirma que los objetivos específicos fueron alcanzados:

- **Modelo de datos y diseño de servicios**
 - El DER definitivo cumple hasta 3FN en las entidades críticas (Jobs, Match) y quedó particionado en tres bases independientes (Users, Jobs, Chat), reforzando el principio una base de datos por servicio. Asimismo, se delimitaron seis microservicios con contratos REST descritos en OpenAPI y límites de contexto claros, aplicando DDD y admitiendo escalabilidad horizontal.
- **Construcción e integración**
 - Los seis servicios fueron implementados con Django REST Framework, contenerizados mediante Docker y orquestados en GKE bajo un API Gateway protegido por Keycloak y JWT.
- **Verificación de calidad**
 - Las pruebas de Capítulo 4 confirman que la solución entrega un servicio confiable y eficiente para los flujos de autenticación, publicación, postulación. Las métricas alcanzadas validan plenamente los objetivos de rendimiento y calidad definidos, lo que deja al Back-End técnicamente apto para su operación real y para una evolución sin fricciones.

5.3 Alcances del Trabajo

El principal logro del proyecto fue el diseño y desarrollo de un sistema Back-End robusto, capaz de manejar los flujos esenciales de la plataforma, como la gestión de usuarios, la publicación y postulación de empleos, y la generación de notificaciones. La implementación de una arquitectura de microservicios no solo permitió dividir de forma clara las responsabilidades de cada componente, sino que también creó un entorno que facilita futuras ampliaciones del sistema, como la incorporación de nuevas funcionalidades o el escalamiento horizontal para manejar mayores cargas.

Desde una perspectiva técnica, este trabajo contribuyó al fortalecimiento de las prácticas de desarrollo de software distribuido, integrando herramientas modernas como contenedores Docker y simulaciones de carga con Apache JMeter. Además, el enfoque modular facilita la implementación de mejoras progresivas y la adaptación a nuevos requerimientos que puedan surgir con el tiempo.

5.4 Limitaciones del Proyecto

Si bien el sistema desarrollado cumplió con los requisitos funcionales y de rendimiento esperados, se identificaron algunas áreas que podrían beneficiarse de un trabajo futuro. En particular:

- **Notificaciones en Alta Demanda:** El Notification Service mostró ligeros retrasos bajo cargas extremas. Optimizar este componente utilizando colas de mensajes, como RabbitMQ o Kafka, podría mejorar su desempeño.
- **Evaluación Integral:** La validación del sistema se realizó principalmente en un entorno de pruebas controlado. Una evaluación más prolongada en condiciones reales de operación proporcionaría información valiosa sobre su desempeño en producción.

5.5 Contribuciones y Aportes

El proyecto StuTrabajo genera un impacto significativo al crear un puente entre estudiantes y empleadores, facilitando la generación de ingresos para los estudiantes y promoviendo su continuidad en la educación superior. Desde un punto de vista técnico, este trabajo:

- Introduce una arquitectura escalable y moderna, aplicable a otros sistemas con requisitos similares.
- Proporciona una base sólida para el desarrollo de plataformas tecnológicas destinadas a resolver problemáticas sociales.
- Destaca la importancia de un diseño modular y la validación exhaustiva en proyectos complejos.

5.6 Recomendaciones y Líneas Futuras

Para quienes deseen profundizar en este trabajo o construir sobre los resultados alcanzados, se sugieren las siguientes líneas de acción:

- **Ampliación de Funcionalidades:** Incorporar módulos adicionales, como un sistema de evaluación más detallado para empleadores y estudiantes, o un análisis avanzado de estadísticas de uso.
- **Mejoras en Escalabilidad:** Implementar herramientas de monitoreo en tiempo real (como Prometheus y Grafana) para supervisar la infraestructura y detectar problemas antes de que afecten a los usuarios.
- **Validación en Producción:** Desplegar el sistema en un entorno real con usuarios activos y recopilar métricas de uso para realizar ajustes basados en datos concretos.
- **Exploración de Nuevas Tecnologías:** Evaluar la integración de inteligencia artificial o algoritmos de machine learning para optimizar la asignación de empleos y mejorar la experiencia del usuario.

5.7 Reflexión Final

El desarrollo del Back-End de StuTrabajo representó un desafío técnico y conceptual que permitió aplicar conocimientos avanzados en ingeniería de software, al tiempo que generó una solución con un impacto tangible en una problemática social relevante. Este proyecto no solo demuestra la viabilidad de las soluciones tecnológicas en contextos educativos, sino que también resalta la importancia de un diseño bien estructurado y una validación exhaustiva para garantizar el éxito de sistemas complejos.

A través de este trabajo, se ha sentado una base sólida para futuros desarrollos, destacando la relevancia de las plataformas tecnológicas como herramientas para enfrentar desafíos sociales y mejorar la calidad de vida de los usuarios involucrados. StuTrabajo no solo es una solución tecnológica, sino un paso significativo hacia la promoción de la equidad y la continuidad educativa.

REFERENCIAS BIBLIOGRÁFICAS

- Newman, S. (2015). *Building Microservices: Designing Fine-Grained Systems*. Sebastopol, Estados Unidos: O'Reilly Media.
- Apache Software Foundation. (n.d.). *Apache JMeter*. Recuperado de <https://jmeter.apache.org/>
- Postman. (n.d.). *Postman API Platform*. Recuperado de <https://www.postman.com/>